Karlsruher Institut für Technologie

Linking SEC XBRL Data With Yahoo! Finance Data for Online Analytical Processing

Abschlussarbeit
von
Tobias Weller

An der Fakultät für
Wirtschaftswissenschaften des KIT

In dem Studiengang
Informationswirtschaft

eingereicht am 08.05.2013 beim
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren
des Karlsruher Instituts für Technologie

Referent: Prof. Dr. Rudi Studer
Betreuer: Benedikt Kämpgen

Eidesstattliche Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Ort, Datum	Unterschrift

Table of Contents

1 Introduction

The semantic web is a concept created by Tim Berners-Lee and can be seen as a further development of the World Wide Web. [GeCh03] Within this concept, the data is interlinked even across different data sources. This enables the creation of a large knowledge base, which is accessible for users and machines. Each resource, regardless if the resource is a real-world object or a fictional figure, can be described using semantic web technologies. [HKRS08 P. 38] Multiple languages exist to describe the resources in a semantic context e.g. OWL and RDF(S). RDF is a language, which is used to represent information based on conceptual graphs. There are different syntax forms to represent the information in the RDF graph e.g. N3 or RDF/XML. However the information of a certain RDF graph does not depend on the syntax form. N3 was build to provide an easier human-readable and compact serialization than RDF/XML. [FFST11] Although RDF is suitable for representing facts in different representations, some facts cannot be represented easily. N-ary predicates, which are used e.g. in nested statements, cannot be expressed within a single statement. Reification is needed to describe the statements in a RDF representation. Within the reification process, auxiliary nodes will be inserted in order to represent the statements in RDF, so the RDF representation is more complex. [cf. HKRS08, P. 80] OWL is based on RDF and offers more classes and properties to represent the information. Due to the higher number of available constructs, more complex relationships and degrees of abstractions can be modeled. An advantage of representing information in semantic languages, like RDF, is that the information can be read and understood by human and machines. [HKRS08, P. 12] Machines can understand the information and semantics of the data. Furthermore, the machines can handle the information and create out of the given information new knowledge, which was not explicitly given. Also machines can check by using logical functions and rules whether the knowledge is consistent or not. The new information, created out of the given, is called implicit information. So-called reasoners fulfill the process of self-closing by checking logical contexts. The result is a grown knowledge base with more information than before. [HKRS08, P. 11 f.] Another advantage of semantic technology is the possibility to interlink information in an easy way with other data sources. By the combination of machine-readable languages and interlinked data, clients can jump between the information of different data sources, which are linked, and collect information to all kind of topics. The plan of this research was to apply semantic web technologies for financial analysis.

Financial analysis evaluates the economic situation of companies regarding their future profit and liquidity. [GABL13] The data for financial analysis is based on balance sheet data, annual reports and external factors e.g. trends of a business sector. The financial analysis can be classified into an internal and external financial analysis. If only public available data e.g. balance sheets, stock market data or business reports are available for the financial analysts, the analysis refers to the external analysis. In

contrast, when a company provides internal data from the cost accounting, the analysis refers to the internal financial analysis. [Fisc05] Bank employees and auditors will mainly perform the external financial analysis. According to the authors Klaus Spermann and Patrick Scheurle [SpSc10], there are the following main tasks for financial analysis, which are:

1. Data collection
2. Calculating and analyzing the data
3. Make forecasts

Applying semantic web technologies to financial tools can help analysts to achieve the aim of analyzing companies. Creating a knowledgebase, where comprehensive and interlinked data is published, facilitates the collecting of data due to the fact that the data is not scattered in the web. Also new information and circumstances can be discovered by using logical reasoners to support analysts making their decisions. The U.S. Securities and Exchange Commission issued rules requiring companies to provide financial statement information in the XBRL interactive data format. This opens a completely new field of mechanics and analysis of data. Different types of fillings are available at the SEC database. Among others, 10-K filling for annual reports and 10-Q reports for quarterly reports are available. The SEC XBRL data serves as a fundamental data source for our financial system.

At the beginning of this thesis, some background knowledge will be conveyed. The SEC and the published fillings will be presented. The published fillings will represent one data source of the financial information observation system. Also the basic concepts of linked data and OLAP will be explained in this chapter. Chapter 2 - Motivation and Research Questions, contains the reasons and objectives for this thesis. After that, related works, which have similar objectives according to a financial analyzing system, will be considered evaluated in chapter 3 - Related Work. The evaluation of related works helped us to assess our own financial system, addressed new ideas and prevented errors. Then, in chapter 4 - Implementation FIOS 2.0 the realization of the FIOS[1] system will be explained. For this, the chapter is separated into three parts. The first part describes the requirement for an integrated financial system. The second describes the integration of SEC XBRL and Yahoo! Finance Data. In this chapter we will explain the particular ETL processes of the FIOS 2.0 system. At the end of this section, a short review will evaluate the results made in this thesis. Through the participation in a challenge with the FIOS system, the results and feedback from this challenge help us to evaluate the system. Besides this, we will also evaluate the system according to the requirements made. Finally, a conclusion on the end of this thesis will summarize the results.

[1] Financial Information Observation System

1.1 The U.S. SEC and XBRL

The U.S. Securities and Exchange Commission (SEC) is an American federal agency established in 1934 by Frank D. Roosevelt. Their main purpose is to regulate the market. [USSE13] The U.S. Securities and Exchange Commission issued rules requiring companies to provide financial statement information in the XBRL interactive data format. Investors should therefore have the chance to inform about companies and sectors before they do investments. This should prevent fail investments that would lead to possible bankruptcy. These submitted documents contain financial information like balance sheets, operating data like net sales and a preview of the planned investments of the company. An advantage of the published fillings is the transparency of companies due to the periodical submission of financial data. Wrong decisions and changes in the enterprise value can now be checked and tracked back. The documents have to fit a special filling type, depending on the information, which they brought. For quarterly financial statements the form name is 10-Q and for annual Reports 10-K. There are many other forms e.g. the 8-K for announcements of corporate changes, which are not discussed in further detail here. A complete list of filling types and their purposes can be seen here.[2] The underlying language of these form types is XBRL[3]. The basic concept of XBRL builds XML, a well-known web language for exchanging information. XBRL represents an extension to this by adding taxonomies for describing business data and hierarchies. [DFOP09] It was developed by Charlie Hoffmann in 1999 and modified in the year 2001 to fit the W3C recommendation XML. [StCa10]. The language is suitable for exchanging, reporting and analyzing data due to the standard it uses. Furthermore, the used XML standard and the compatibility with XML schemas allow an easy interchanging. Beside this, XBRL is open source and very flexible due to the extensibility it offers. This enables the adaption to nation regulation and laws. Also specific financial data of each business sector can be mapped in XBRL. So information can easily be exchanged, viewed and analyzed across countries and different business sectors. By using only one standardized business language, costs can be reduced due to the fact that only one format is used instead of a high number of formats. Moreover, XBRL allows versioning, and so the language can be enhanced and accommodated to recent developments. As another advantage of XBRL, the authors of “XBRL For Dummies” mention that XBRL information can easily be transferred to spreadsheets. [WeWi08] An exemplary section of an XBRL file is given in the figure 1.

[2] http://www.sec.gov/info/edgar/forms/edgform.pdf, 7th May 2013
[3] eXtensible Business Reporting Language

Section of an XBRL document:

```
<us-gaap:AccruedIncomeTaxes contextRef="BalanceAsOf_31Dec2010"
    unitRef="USD" decimals="-6">297000000</us-gaap:AccruedIncomeTaxes>
<us-gaap:AccruedIncomeTaxes contextRef="BalanceAsOf_31Mar2011"
    unitRef="USD" decimals="-6">113000000</us-gaap:AccruedIncomeTaxes>
```

Fig. 1: XBRL instance from Metlife Form 10-Q, 31.03.2011 [RCH12]

An XBRL document consists of two parts, named XBRL instance and XBRL taxonomy. While the XBRL instance contains primarily the business information, the XBRL taxonomy contains information about the calculations of the key figures, definitions and labels in different languages. A XBRL instance starts with the tag "<xbrl" followed by the business facts. Business facts contain the relevant number, which will be reported, and the number of decimals. Additional information like the unit, footnotes and context information can be added as XML attributes to the business fact. Each fact is connected to a concept. Concepts are part of the XBRL taxonomy. The concepts define the tags and contain detailed information like definitions or formulas used in the XBRL instance. These concepts are centralized in the XBRL taxonomies, which are commonly stored in another document, instead of including it in the same document. Due the centralized description of the concepts used in the XBRL instances, each country can publish its own XBRL taxonomy for the business sectors that fits to the needs of these sectors. Depending on the business sector or industry in which a company is working, balance sheets will have different styles and will use different calculations for their key figures. [HoWa09] When regulation changes, not the whole XBRL document has to be changed, only the taxonomy will be adapted to the new regulations. GAAP[4], created by the Financial Accounting Standards Board, presents next to IFRS[5] an international accepted taxonomy and is of primary importance due to the fact of the regulated use by the SEC. Figure 2 shows the interaction between the XBRL components. [HoWa09]

[4] Generally Accepted Accounting Principles
[5] International Financial Reporting Standards

Interaction of the individual XBRL elements:

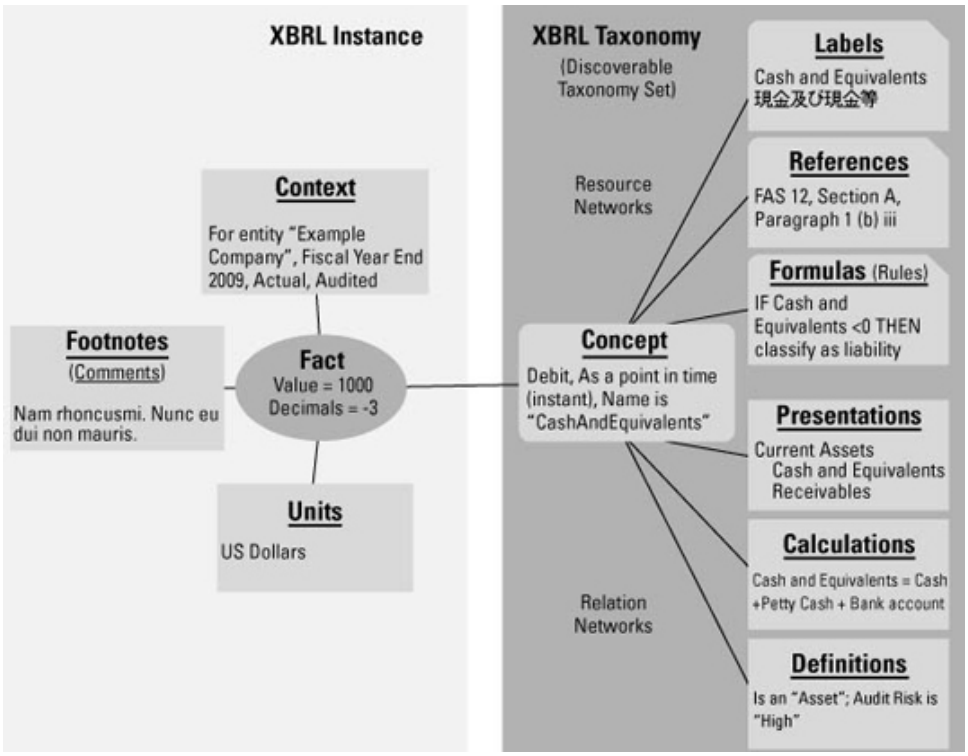


Fig. 2: XBRL Components [HoWa09]

XBRL is amongst others recommended by the European Banking Authority.[6] By making the financial data of companies publicly available, a complete new possibility of analyzing data is opened. One possibility, which is used by the FIOS system, is Semantic XBRL. Semantic XBRL are XBRL documents, which are converted RDF. Data from the XBRL documents can be linked to other semantic data to take profit of the new gained data, as Roberto García and Rosa Gil did, by transforming the XBRL data into RDF. [Wood10]

1.2 Linked Data

The World Wide Web offers users the possibility to publish information across the web. However, the published data are scattered in the web. Tim Berners-Lee introduced in his web architecture note the linked data principles for an interlinked and structured data on the web. [Bern06] The principles are listed in figure 3.

[6] http://eba.europa.eu/Supervisory-Reporting/XBRL.aspx, 7th May 2013

Linked Data Principles:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
4. Include links to other URIs. so that they can discover more things.

Fig. 3: Linked data principles by Tim Berners-Lee [Bern06]

The first principle aims at identifying things. Both, real-world entities like persons, buildings or animals and abstract concepts, e.g. a character in a novel, should be given a unique name in the form of a URI[7] to identify them. The second principle strengthens the first by giving the entities a HTTP[8] URI. This will enable clients to look up the URI and request a description of the resource. For making URI dereferenceable, there are two strategies on which we would like to focus only briefly. The first is 303 URIs. In this strategy, the server receives a HTTP protocol and answers not with the real-world object itself, but with the HTTP response code "303 See Other" and the corresponding URI. By following the URI, the client receives a web document that contains information about the requested object. The main criticism of this strategy is that two HTTP requests are necessary to retrieve a single description. The second strategy, the Hash URIs strategy, avoids this critic by using hash tags. In this strategy, the URI contains a special part after the hash tag. This part is called fragment identifier. However, the fragment identifier will not be used to retrieve a web document, but only the base URI. The document contains the information about the object in the fragment identifier, but can also contain additional information about other objects. The advantage, compared to the 303 URI strategy is the reduced number of HTTP calls. However, when retrieving a web document with the hash tag strategy, the whole document will be retrieved which contains all information that has the same URI. Both strategies are used for making HTTP URIs dereferenceable. The third principle says that standards should be used for providing information, so everyone can use the information. A standard like RDF is both, simple and very powerful. RDF can represent a huge number of information and facts and has also different kinds of serializations for representing information. The last principle says that information in a document should be linked to information of other documents in the web, so that clients can follow the links and retrieve all over the web information from different data sources about real-world entities or abstract concepts. [HeBi11] By using these four rules, the information in the web is not scattered anymore. Clients can access them and follow the links to other objects to retrieve information. The popularity of linked data can be seen in figure 4. This shows the

[7] Unique Ressource Identifier
[8] Hyper Text Transfer Protocol

growth of linked data sources in the web. The left image shows the available data sources on the 10th of November 2007 and their connections. In total 12 data sets were available. The right image shows the Linked Data Cloud on September 2011. The number of available data sets is 295 and they are grouped according to the information, which they provide. Information provided by the government is on the left, colored in dark green. In this category, the SEC database is included as well. Richard Cyganiak and Anja Jentzsch collected the data for the Linking Open Data Cloud.[9]

Compared Linked Open Data Cloud over the year:

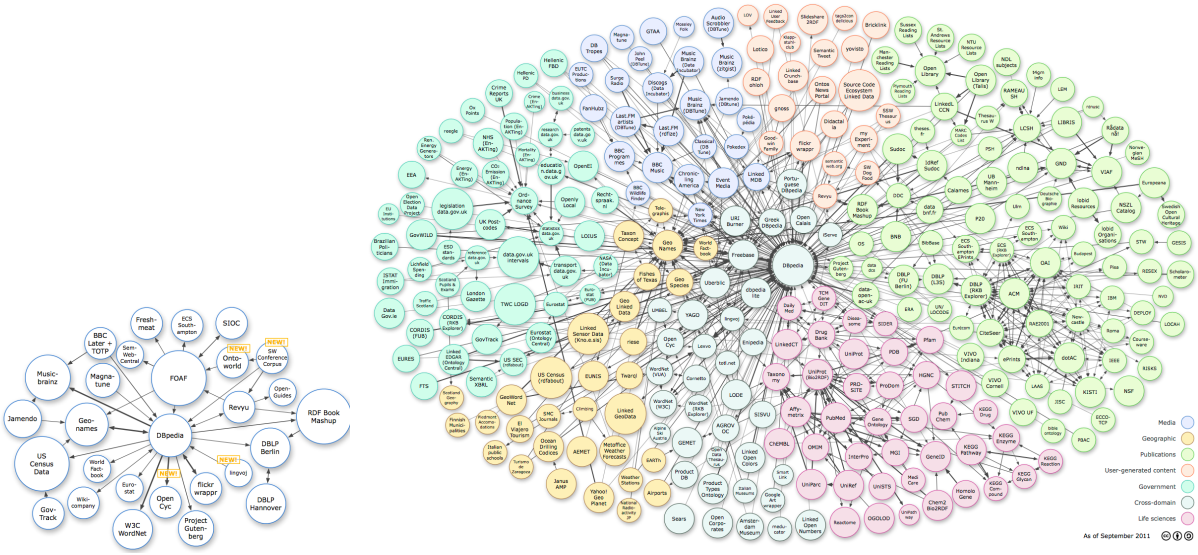


Fig. 4. Available data source for the year 2007 and 2011.

1.3 OLAP

OLAP[10] is a database technology used for complex analysis. It enables the multidimensional view of the data and executes them in different manners. The used structure for enabling OLAP operations is a cube. A cube represents a closed data pool of a business sector. The data will be arranged logically in the cube structure. Cubes consist of multiple dimensions that describe the data and allow a grouping of one or more levels. For analyzing data in the cube, some basic functions are available. Among others the function slicing, which means the filtering for a special dimension, is available. Another example is the Roll-Up operation that consolidates the data on a higher aggregated level. [Vass02] A special aspect for analyzing data is the possibility to represent hierarchies with OLAP. So OLAP functions can also follow hierarchies to enable the selection of the data on different levels of the hierarchy. An exemplary

[9] http://lod-cloud.net/, 7th May 2013
[10] Online Analytical Processing

Cube can be seen in figure 5. This figure is a representation of the multidimensional model of a cube from Benedikt Kämpgen and Andreas Harth. [KäHa12]

Cube representation:

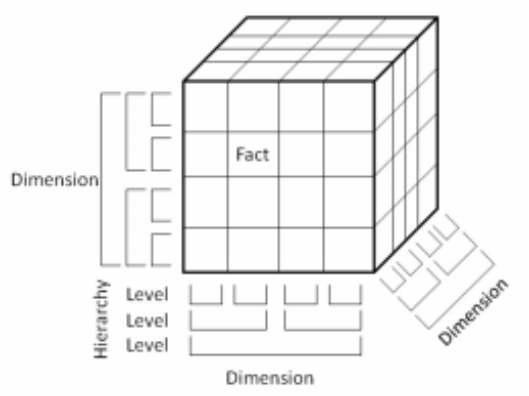


Fig. 5: Common Multidimensional Model [KäHa12]

OLAP is widely used and has a wide range of basic operations to analyze data, it is also suitable for financial systems. However, so far no semantic data model "…has become generally accepted due to their respective deficits and the lack of support from the data warehouse industry"[MW02P]. To enable the use of OLAP functions for semantic data, a special data model is needed. The data model RDF Data Cube Vocabulary will be introduced in chapter 4.1.3 – Data Model. For displaying and executing the OLAP functions the FIOS system uses among others Saiku, an open source analysis tool. Saiku will be introduced in chapter 4.2.2.6.3 – Saiku.

2 Motivation and Research Questions

The Internet is full of data that can be accessed at any time and from almost everywhere. People or organizations can upload their own data and provide this data to other people, however the data is not in a common format. The presentation of the data depends on the publisher, because he decides in which format he would like to publish the information. Besides the problem of different formats, the data is also scattered in the web. Information about a certain topic is not combined on a special page or format and thus the finding process and the evaluation of data is made difficult, not at least because of the growing internet.[11] In addition, most of the data in the Internet is not published on a semantic level. The semantic web enables machine-readable analyzing of the data. Machines, that can assimilate the information, validate them to their correctness and generate new information out of the available data. Different information can be published in the web. A special part of the published data for companies, investors and stakeholders are the financial information. They receive a high number of their information from the web. However, the data is not published in a common format. Publishing the financial data in different formats and across the web makes analyzing them harder. First, the data has to be found, perhaps refined and then stored in a common database. Only after those steps, the formerly scattered data, is combined and ready for analyzing. Semantic Web allows to link information, thus the information is not scattered anymore. Another problem of various data formats is that the exchange of the data is more difficult, because programs perhaps use different data formats and thus can partially or completely do not handle the formats. Due to the increasing globalization of companies, the exchange of data must be secured. Among others, business data from subsidiaries or other agencies will be transmitted. Often, these subsidiaries are in different jurisdictions where other regulations exist. The introduced computer language XBRL demonstrates a format for exchanging business information in one common format, which can be adapted for every jurisdiction and thus significantly facilitates the exchanging of business data. [WeWi08] However, the business data in the XBRL format cannot be linked to business data from other data sources. Applying semantic web technologies to XBRL data enables the possibility to link the information to other information in the web. Thereby new possibilities in analyzing financial data are given by considering multiple information from different sources. By applying semantic web technologies to financial data, several advantages are given. The integration of data between companies and communities facilitates the exchange of information and enables insights in different sectors. Another advantage is given by linking information to other information across the Internet. By linking information from different data sources a network of financial facts and information is

[11] http://www.business-standard.com/article/technology/internet-grows-to-over-252-million-domain-names-in-2012-113040900185_1.html, 7th May 2013

generated. This network can be used to analyze integrated data that were formerly separated.

In the year 2012 Andreas Harth, Benedikt Kämpgen and Sean O'Riain took part in the XBRL Challenge. The XBRL Challenge is a contest proclaimed by XBRL US, a national consortium to support the promotion of XBRL. In this challenge individuals or teams submit own created tools or programs that enables users to analyze data that rely on XBRL data from the SEC. The judges, consisting of CEO[12], developers or managers, evaluate the submissions and honor the winner relating certain criteria. Based on the results in the XBRL Challenge of 2012, we continued by creating a high degree of transparency and exchanging financial data.

The goals of this thesis are the proceeding of making data from different data sources available in RDF format, so the information published in the data sources can be interlinked with other data. Thus an information network is formed that follows the linked data principles. The developed network of information, which results by connecting the different data, allows the exploration of information from different data sources. However, one of our objectives is not only to publish data in a RDF file, but as well to combine different data in one database. Every information in the database should be crawled from public available data sources and not maintained by our own. To make the linked data available in our database we also show how to use the data to analyze financial information, which is published according to the linked data principles, in a manner that also business analysts can use the system. Therefore we would like to show how the field of research "Linked Open Data" could be applied in financial institutions, so that they can take profit out of it. A further goal of this thesis is that the developed system should be generic, so that the financial system is not limited to a certain data source or a certain server. These objectives had been taken into account while the implementation of the FIOS system.

[12] Chief Executive Officer

3 Related Work

In this chapter papers and works in the sector of analyzing XBRL data, from companies or universities will be introduced. The analyzing functions and possibilities for business analyzers, that the related works offer, will be considered. Particular attention will be paid to the data integration. The data integration is important as it enables a larger selection of data and thus analyzing possibilities. Also it enables the possibility to analyze data from different data source within one system.

3.1 XBRL to Spreadsheet Converter

Developed by ConnectCode Pte Ltd., the XBRL to Spreadsheet Converter is a free available tool for analyzing XBRL data. However, the tool was not available while writing this thesis. We refer to a published paper. [GoMS12] Necessary is Microsoft Excel 2002 or later with macros enabled. The Excel document contains multiple tabs. Users can enter the location of a XBRL instance document and extract it into the Excel document. The corresponding information will be displayed in Excel. Multiple years can be extracted. Each will be added in a column. However, users need to know the URL of the XBRL documents, so that the data can be downloaded into the spreadsheet. First of all, they have to look up for the corresponding document in the SEC database and copy the URL into the Excel file. Comparing this process to the FIOS system, which has an automated ETL process, this step is laborious. When closing the Excel sheet, the downloaded information is lost. When reloading the application, users have to extract the XBRL information again for analyzing them. XBRL documents can be extracted for multiple companies. This enables analyzing multiple companies for multiple years. Within the Excel sheet, all functionalities of Excel are available, so users can use them for calculating KPIs[13], calculating the percentage change of a key figure or display the information in charts.

Compared to FIOS, the XBRL to Spreadsheet Converter can only display XBRL documents. Data in other formats cannot be displayed. This tool is limited on XBRL instance documents and thus analyzing possibilities are also limited, because information from other data sources cannot be included. If at all, information from other sources has to be merged manually or via a program. Furthermore, users have to have at least some basic knowledge in Excel. The tool by itself is easy to understand indeed it just lists the extracted information. But for calculating KPIs or the percentage change of a key figure, basic Excel skills are necessary. Also the use of Microsoft Windows is stringently required. The correct representation of the data in other operating systems, like Mac OS, is not guaranteed. The different descriptions of the balance sheet items of the companies exacerbate analyzing the items. Companies for

[13] Key Performance Indicators

example, use partly different names for “sales revenue”. The extracted information is not rehashed in the Excel file. The information of the balance sheets is just listed in one column. The possibility of analyzing the data with OLAP functions is not given. By only extracting and listing the information, a data model is not given. The main advantages of the XBRL to spreadsheet converter are on the one hand the easy and understandable user interface and on the other hand, the possibility of the specific analyzing XBRL instance documents. However, comparisons of multiple years and multiple companies will get fast confusing.

3.2 Calcbench.com

Calcbench.com[14] is the winner of the XBRL Challenge 2012. It is an online toolkit for analyzing XBRL data. By entering a ticker symbol of a company, users can select and view reports from this company. It is possible to view quarterly and yearly reports. Analysts can view the XBRL data and compare them to the data of the company from former quarters or years. This enables year-to-year comparisons of balance sheets. Percentage changes of the balance sheet items will automatically be calculated as well as, in some cases, forecasts to future values. Not only year-to-year comparisons for one company are possible, but also comparisons between multiple companies. There is no restriction on the number of compared companies. By clicking on a balance sheet item, a line chart appears that shows the temporally changes of this item. The strengths of Calcbench.com are mainly due to the comparisons between companies and years. The displayed percentage changes of the balance sheet items enable the fast recognition of huge changes. Furthermore users can add and save annotations to each item. Thus users can write down their thoughts directly to the system while analyzing data. On a later request, the annotations appear for the report again. It is also possible to create workgroups and share the analysis a user did to the others within the workgroup. Therefore a community can be created where every user realizes its knowledge. Calcbench.com can therefore become a knowledge base. This knowledge base is limited to the annotations that the users make on the balance sheets data. Realizing those annotations in a refined form is not the main attention of Calcbench.com. There is also the possibility to display the data in spreadsheet view. Users can add or delete rows; also basic math functions, like sum, max or min, are available. But these functions are very limited. Highlighting data with a bold script or changing the foreground color enables user to mark special data. However, the spreadsheet view is not yet sophisticated. The limited number of math functions, where e.g. multiplication or division is not available, and the long response times by changing data, shows that this view is not yet sophisticated. A function, for exporting the data to a local computer, to enable users to use those information for calculating

[14] http://www.calcbench.com/, 7th May 2013

key figures, highlighting and refine them and finally uploading their results again to Calcbench.com would fit more.

However, Calcbench.com is restricted to XBRL data from the SEC. Comprehensive comparisons to other data, e.g. stock data, are not possible. This limits the possibilities of analyzing data due to the fact that analyzing only balance sheets data is only one aspect of changed balance sheet items. Also aspects from outside must be considered. These aspects may be e.g. governmental restrictions on a special sector, NASDAQ[15] changes or political changes in a country. Users of Calcbench.com can thus consider only one view of the data without considering others. Calcbench.com is hence slanted toward XBRL data without involving other sources to extend the possibilities for analyzing data.

Calcbench.com is a very good tool for analyzing XBRL data in particular or in a small workgroup. The clearly represented vision relieves analyzing the balance sheets data. The partial considering is certainly a deficit that can be mend by adding a new data source. Thus users can involve more information to their analysis and make them more accurate.

3.3 CorpWatch

CorpWatch[16] participated in the XBRL Challenge 2012. They created an API[17] for the SEC Data. An API is a program interface that is used by software systems to communicate with other programs. CorpWatch provides with their API a search interface to look up information located in the company fillings, published at SEC. Company information includes company name, former company name and/or address if existing, address, country, industry code (SIC), tax number and subsidiaries. This information is located in SEC 10-K and exhibits 21 fillings. But the API only accesses those fillings that have been uploaded into the CorpWatch database and parsed successfully. The program does not involve more recent fillings. However, the information provided by the CorpWatch API may be obsolete and thus it is not guaranteed that the information is correct. It may happen that the address of a company changed and updated fillings have not yet been loaded. Furthermore, CorpWatch do not access directly the SEC database. They use their own database, where the downloaded and refined 10-K and 21 fillings from SEC rely[18]. According to this, it may be possible to load new fillings into the database. Due to the non standardized format of the Exhibit 21 fillings that are published by the SEC, parsing becomes a problem. Errors while parsing may happen and thus the correctness of the data is not guaranteed. Within Exhibit 21 fillings contain subsidiary information like

[15] National Association of Securities Dealers Automated Quotations
[16] http://www.corpwatch.org, 7th May 2013
[17] Application Programming Interface
[18] http://api.corpwatch.org/documentation/source/README, 7th May 2013

company name. An example of an Exhibit 21 filling can be seen on the SEC homepage[19]. According to tests, CorpWatch did, they believe that 90% of the subsidy companies are correct[20]. CorpWatch publishes the information about a company in XML form.

Compared to the FIOS system, the CorpWatch API only provides company information and no possibility of analyzing them. The information, they offer is non-financial information. Compared to the Edgar Wrapper of the FIOS system, the CorpWatch API also provides address data and company name data and furthermore subsidiary information, but no financial information, like total asset or goodwill, like the Edgar Wrapper does. Furthermore, the Edgar Wrapper publishes the information according to the Linked Data principles. Information is linked with other information, even in other data sources like freebase or DBpedia. CorpWatch do not offer such links. However, XBRL is based on XML thus no big transformation of the data is given. The API tends more to the use in different application. One example is Croctail[21]. Croctail offers information, based on the CorpWatch API, concerning companies and their subsidiaries. Users can browse through the provided information and get Address data, company name and also sectors of companies. By connecting Google maps widget, the subsidiary locations can be displayed on a map. The popular XML format facilitates the treatment and use of the information from the former XBRL documents. The Edgar Wrapper does not offer geographical information about the companies.

The constriction on fillings manually loaded, as well as the constriction on meta-information like the company name or address, is not sufficient for analyzing business data. It is no tool for business analysts, because it does not offer a possibility to see balance sheet data. Therefore it is not suitable for analyzing XBRL data.

3.4 Sector3 – XBRL in the 3rd Dimension

Sector3 is developed by Phil Crowe and Jim Truscott, which participated in the XBRL Challenge in 2012 with their entry “XBRL to XL”. They refined their approach of transforming and displaying XBRL data in Excel and participated with Sector3 at the XBRL challenge 2013. Sector3 is part of XBRL to XL and enables meaningful analyzes of sectors. Users can look up for fillings and select up to five fillings. The limitation on five fillings is thus constituted that the processing and downloading time does not take too long time. However, more companies can be added by downloading further Excel files with the corresponding fillings and merge them with other downloaded excel files. The fillings in the Sector3 database contain each filling from

[19] http://www.sec.gov/Archives/edgar/data/45012/000004501213000086/hal-12312012xex211.htm, 7th May 2013
[20] http://api.corpwatch.org/documentation/faq.html, 7th May 2013
[21] http://croctail.corpwatch.org/, 7th May 2013

the SEC database and will be refreshed every 10 minutes. Selectable are both 10-Q and 10-K fillings. This approach is similar to the XBRL to Spreadsheet converter. However, the selection of the fillings occurs in the web with a user-friendly search box. With Sector3 users can classify the segments either with the standard sector from the standard code or with their own expressions. After classifying the segments the Excel file is ready for downloading. It contains the information of the fillings in a refined way for using them in a pivot table. The spreadsheet also contains several sector and company summaries, based on pivot tables. Included filters allow the selection of specific measures. However, also company information is displayed, Sector3 is concentrated on analyzing business sectors, not companies. This becomes apparent by regarding the permanent grouping of the results in the sectors of the given pivot tables. A comparison of companies in a form like the XBRL to spreadsheet converter provides can manually be created by a pivot table. However users need to have profound experience and knowledge of Excel. Users without such experiences can hardly answer complex business questions. Sector3 is concentrated on analyzing and displaying XBRL data. Additional data from other data sources can added in the Excel file. However, the data have to fit to the data scheme, and so a manually refining step is possibly needed to include external data for analyzing purposes. Therefore analyzing external data is not easy and is concentrated on XBRL data. Tests also showed that Sector3 has problems converting the XBRL data into Excel. [Trus13] The issue is that in some cases companies do not use the official US-GAAP rules for publishing a balance sheet item, but use their own extension. Companies introduce own extensions if the US-GAAP does not offer corresponding balance sheet items. This leads to mistakes in calculating key figures and ratios in the Sector3 program. The problem can be solved by manually changing the extension to the official US-GAAP.

3.5 Brief Summary

We have seen different approaches of displaying and analyzing XBRL data. Most of those programs present the data similar to Excel. The decisive factor of the FIOS system is the representation of the data according to the Linked Data principles and thus the easy linking of new external data sources. The FIOS system is not limited on XBRL balance sheet data. Additional data can be included that follows the Linked Data principles. The SPARQL endpoint of the FIOS system allows the access of additional programs that can handle SPARQL requests. If not, a middleware is needed to transform the requests and to return the corresponding answer from the endpoint, as Olap4ld does. While the represented related works only observe data from one data source, the FIOS system combines data from different data sources. The interlinked and integrated data, combined with the easy access of additional data sources is the strength of the FIOS system.

4 Implementation FIOS 2.0

The implementation of the FIOS system is separated in three parts. The first part focuses the requirements for an integrated financial analyzing system. Therefore business questions are discussed which later should be solved in our financial system. The selection of data sources, by which the data will be linked with the SEC XBRL data, is discussed in chapter 4.1.2 - Selection of Data Sources. Therefore we will analyze different data sources according to the number and kind of available key figures, data quality and other quality features of a data source, to select the best possible data source to answer the business questions. After the selection of the data source, a suitable data model is needed to represent the data. This topic is part of section 4.1.3 - Data Model. After section 4.1 – Requirements for Integrated Financial Analysis follows the integration of SEC XBRL data with Yahoo! Finance Data. Therefore we will first describe the architecture of the FIOS 2.0 system and see the correlation of the different parts of the system. Then in chapter 4.2.2 - Automation of ETL Process, the single modules will be introduced and described in detail. These modules are necessary to allow the automation of the ETL process. The evaluation of the FIOS 2.0 system is carried out in section 4.3 – Evaluation: FIOS 2.0 for the XBRL Challenge 2013. Therefore we will on the one hand evaluate the competition in the XBRL Challenge 2013, where we took place with the FIOS 2.0 system, and on the other hand discuss the results with the requirements made in section 4.1 – Requirements for Integrated Financial Analysis. Furthermore, the special settings made on the system for the XBRL Challenge will be explained.

4.1 Requirements for Integrated Financial Analysis

One requirement of integrated financial analysis is that the system should be able to answer as much financial questions as possible. Possible business questions will be introduced in 4.1.1 - Business Questions. For answering business questions, data is needed. Since the XBRL data is already available via the Edgar Wrapper, a new selection of data source to answer further business questions will be considered in section 4.1.2 - Selection of Data Sources. In section 4.1.3 – Data Model, a suitable data model will be introduced that fits to analyze financial data. Every data from different data sources should be modeled according to the acquired data model thus data integration is possible. As a requirement, the system should support the user while the ETL process as much as possible. Best would be if the complete ETL process is automated.

4.1.1 Business Questions

Having satisfactory financial data allows analyzing of companies or sectors in a suitable way. However, analysts often have diversified questions e.g.:

- Which company in the computer sector has the highest net income?
- What is the price-to-book ratio for Microsoft Corp. and does this ratio conform to its real balance sheet data?
- How has Microsoft Corp. involved over the last years?

Especially the last two questions lead us to the question if the presented data sources are sufficient for complex analyzes. So far the data sources of the FIOS system were SEC, freebase and DBpedia. Adding a new data source allows asking more questions and by understanding the responses, the progress within the business sector or company can be understood better. An excellent addition to the existing FIOS system would be a data source that publishes stock market data. The stock market data would show an external view of the company while the SEC data shows an internal view. The comparisons can show analysts if a company is probably overrated at the stock market or can lead to better forecasts into the future. According to the FIOS system, business analysts can ask questions according to the progress of companies over time.

In addition, analysts often consider more than one company and compare them to each other as well as over the years. Therefore the financial analyzing system has to represent on the one hand the values in a table similar to a pivot table, so that the data can easily be read, and on the other hand the system should be able to present the data over periods to see the progress of the displayed key figures. A satisfying representation of the data simplifies the analyzing step. Furthermore the last question, mentioned above, leads to the result that multiple key figures shall be able to be

displayed, due to the fact that analysts often consider several key figures at the same time. Besides considering companies, the system should also be able to aggregate the available data to a higher level. The SIC[22], which is a 4 digit unique key, classifies companies to a certain industry. Each SIC is grouped into one Industry Group. Multiple assignments of a certain SIC to different Industry groups are not possible. An Industry Group can contain more than one SIC. The key for the Industry Group consists of the first three digits of the SIC. The Industry Group is also classified in groups, called Major Groups. The key for the Major Group can be read out of the first two digits of the Industry Group. At the top of the resulting hierarchy is the Division. Figure 6 shows an exemplary part of the SIC hierarchy.

Illustration of a part of the SIC hierarchy:

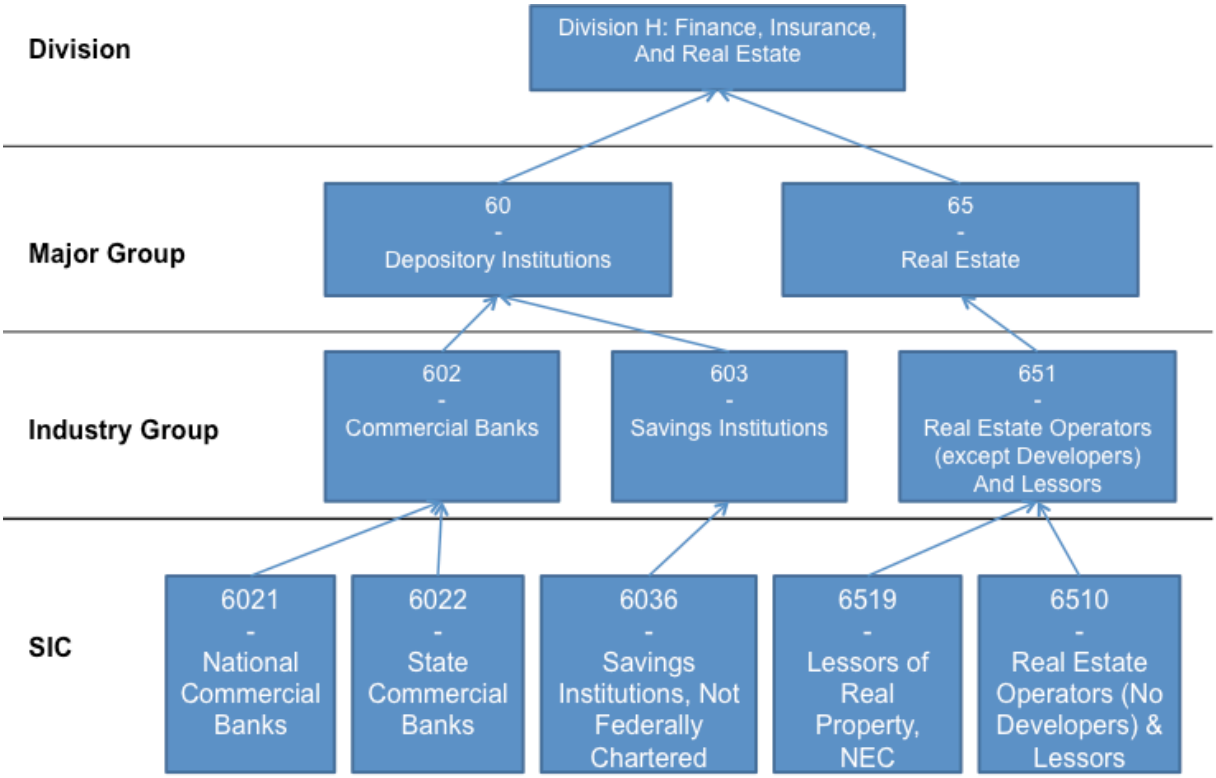


Fig. 6: Example of the SIC Hierarchy

The assignment of companies to a SIC and in turn the assignment of the SIC to the Industry Group and so on is excellent to represent data from different companies on a higher aggregated level. The data of same companies within the same SIC can be grouped and represent the result of the total industry. By using this approach not only financial data for one company can be displayed, but also for a complete business sector.

Another preferable requirement is to have the chance to analyze data according to the geographical allocation of companies. So we can analyze different countries according

[22] Standard Industrial Classification

to the net income, which is generated by the companies in the particular country. It should therefore be possible to answer questions like “Which country generates the highest net income in the automotive sector?”.

4.1.2 Selection of Data Sources

An improvement possibility for the FIOS system was to connect another data source. This chapter deals with the presentation and selection of a new data source for the FIOS system. For selecting a new data source, several criteria were important. The first criterion for a new data source was the free availability. Data sources like EOD data or Reuter were not allowed as a new data source for the FIOS system, because those data sources cost money and do not offer free available data. Also, the data should be available at any time. This must be guaranteed due to the fact that we do not have a database, where we save the data and access them. We only use a wrapper that transforms the data on-the-fly. Another criterion that had to be fulfilled is that the new data source publishes the data in a common format. The format should be easy to access, so that the wrapper can transfer the data fast. An unlimited and easy access is preferable. Also, to include data from other data sources, it had to fit with our data model. As a further criterion, the data source should publish data for a high number of companies and without restrictions on a certain business sector. As the last requirement of the data source, answers to the business questions presented in the previous chapter should be enabled by the resource. For selection purposes we created an Excel file that contains possible data sources and their characteristics. In the following, different data sources will be considered and evaluated regarding the usage in the FIOS system, which is an abstract of the created Excel file. The complete Excel file can be seen in the appendix.

Yahoo! Finance offers two APIs to access stock market data for companies, listed in NASDAQ. The Yahoo! Finance API offers us the possibility to retrieve besides actual data also historical data, even if the number of historical key figures is limited. The historic stock market data is the stock market data that occurs for more than one day in the past. The access to the historic stock market API is granted by the ticker symbol and a time interval. The API retrieves the stock market data for the corresponding company of the ticker symbol within the time interval. The API retrieves only a limited number of key figures. The highest stock value for a certain day and the lowest stock value will be retrieved, as well as the shared volume, the start value and the end value of the stock for the requested day. The other API from Yahoo! Finance retrieves actual stock market data. The entry of an explicit date is not given; the API retrieves the data for the exactly time. Besides the input of a ticker symbol, the short cuts of the requested key figures have to be entered. In total 84 key figures and ratios can be requested. For the Wrapper, the number of key figures is perhaps too much, however by specifying the URL for the API, only the relevant key figures can be requested. A

complete list of available financial data can be seen here.[23] Both APIs publish the financial information in a CSV format. The CSV file allows an easy access for transforming the data into the RDF data model.

Another possible data source is the Google Stock API. The API can be accessed via the ticker symbol of the company. Similar to the Yahoo! Finance API, the Google stock API retrieves NASDAQ values as well. However, compared to the Yahoo! Finance APIs, there is no possibility to access historical stock values with the Google stock API. Google publishes with its API only actual stock market data. In the light of having no own database, where we can save the stock markets, we need for the wrapper at any time access to also historical data. The retrieving of only actual stock market data was a disadvantage of the Google API. Also, the API retrieves only twelve key figures. Compared to the Yahoo! Finance API, which provides 84 key figures, this is a very small number. Furthermore, the key figures in the Google stock API and the Yahoo! Finance API overlap. Each key figure in the Google stock API is also available in the Yahoo! Finance API plus more. However, the access to the Google stock API is very simple. Due to the fact of publishing only actual data, no time interval is needed. Only the ticker symbol of the corresponding company has to be added to the URL. Moreover, the data is published in XML, a very easy accessible format that is recommended by the W3C. Because of the optional access to the information located in the tags and the possibility to represent RDF data also in XML syntax, the transformation to the RDF data model is very simple, which facilitates the conversion process.

The NewYork Times has recently started offering direct access to their database. Multiple APIs are available for accessing information e.g. presidential campaign contribution and real estate listings and sales. After registration, users have free access to the APIs and are free to use them. Also information about companies is available via the semantic API. The API retrieves information in RDF and XML format. The publishing of data in RDF format is useful, due to the fact that the information is already in the Linked Data principle and thus the crawler can access them. The transformation to the data model used for the FIOS system would be simple. The New York Times API contains among others, owl:sameAs links to other data sources, a definition of the company and a counter how often this company has been mentioned in articles of the New York Times. However, the API contains no key figures or financial data of the corresponding company. This makes the use for the FIOS system, whose purpose is the financial analysis with the goal of strong data integration, hard. Also the access to the New York Times data is difficult, due to the fact that a unique ID will be needed. The ID is a unique number in the New York Times database that stands for the corresponding object.

[23] http://www.jarloo.com/yahoo_finance/, 7th May 2013

An overview of the three data sources and their corresponding quality and characteristics can be seen in table 1.

Overview of the data sources:

Data Source	Input	Number of key figures	Max. Delay	Receive format
Yahoo! Finance API	Ticker, Time Intervall	84	15 min.	CSV
Google Stock API	Ticker	12	15 min.	XML
New York Times API	ID	-	-	XML/RDF

Table 1: Overview of the selectable data sources

After considering the different data sources we decided to integrate Yahoo! Finance as a new data source for the FIOS system. The numbers of available key figures are very suitable for analysis and calculating indicators. The published data in CSV format can be transformed easily into RDF. However, actual information means 15 till up to 60 minutes delay to the originally course date. The integration of Yahoo! Finance stock market data and balance sheet data from SEC allows analyzing different data within the same system. To enable the integration of the Yahoo! Finance data, an adequate data model is needed. This will be discussed and presented in the next chapter.

4.1.3 Data Model

A data model is a “collection of concepts, which will be used to describe the structure of a database”[ElNa09]. It is an illustration of the arrangement of the data and describes among others the relationships between the concepts. The data model builds therefore the basis of our database.

Due to the high number of business data in the web, strong operations are needed to analyze them in an efficient way. OLAP enhances functions e.g., slice/dice or drill-down/roll-up for exploring the data in the data warehouse. Commonly the interlinked data is displayed as a multidimensional view of data, called Cube. To use OLAP functions for analyzing the data, we have to apply a well-interlinked conceptual data model. Therefore, we aimed at loading the data from heterogeneous data sources, transforming the data into a semantic data model that fits with OLAP functions and loading them into our data warehouse in which the data is stored for analyzing with OLAP functions. Different data sources publish the data not always in a common data model. Therefore, we had to refine the data into one consistent data model. There are several data models available that allow the publishing of data for analyzing them in a cube format. We decided to use the RDF Data Cube vocabulary (QB). This vocabulary

follows the Linked Data principles and is recommended by the W3C. [CyRe13] SCOVO[24] is a vocabulary to describe statistical data. SCOVO is deprecated and is not maintained anymore. Compared to the SCOVO vocabulary, QB is easier to handle and also better for capturing semantics of the linked data. Furthermore every SCOVO vocabulary can be transferred into the QB and mapping the data sets to the QB occurs via a simple scheme. [KäHa12] By using the RDF data Cube vocabulary, OLAP functions can be used for analyzing the data. Observations will be made for a special issuer at a certain time. Each observation includes the dimensions: dtstart, dtend, subject, issuer, segment and the value. The dimensions will be used to identify the observations. The observations will be grouped into one data set. The data set, indeed, has a certain data structure definition, which describes the structure of the contained observation. Dimensions that are mentioned in the observations of the data set will be defined in this structure. Figure 7 describes the RDF Data Cube Vocabulary with its relations.

Section of the RDF Data Cube Vocabulary:

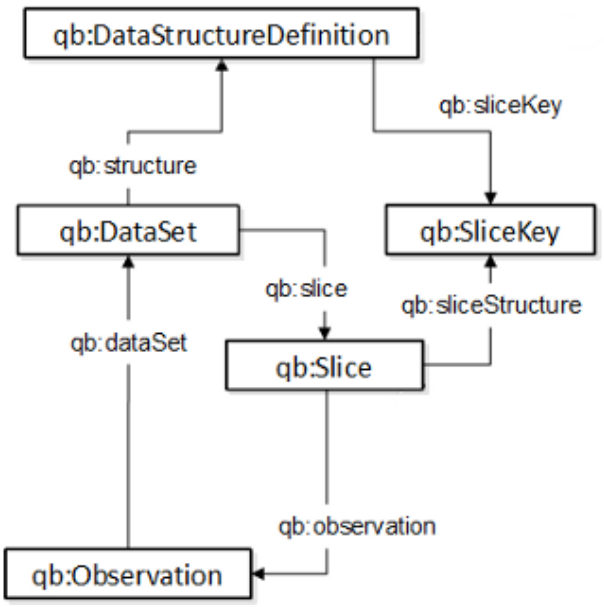


Fig. 7: The RDF Data Cube Vocabulary

To describe companies and their relation to the data sets, we used existing ontologies e.g. FOAF to model the name of the companies as much as possible. In cases, where no ontology are suitable, we insert own created properties e.g. the connection of companies to the archives, where the observations are collected.

[24] Statistical Core Vocabulary

4.2 Integration of SEC XBRL and Yahoo! Finance Data for Analysis

This chapter describes the implementations of the FIOS system. As data sources serve the SEC XBRL data, Yahoo! Finance data and other freely available data. First of all the architecture of the FIOS 2.0 system will be introduced to get a better knowledge about the correlations between the different parts of the system. After making the architecture known, the ETL process and its single steps will be introduced. These steps help to integrate the financial data, so that they can be used to analyze them. Data integration indicates the merging of data from different data sources and the uniform representation of them. [cf. HGHM11]. The data integration is therefore a crucial issue of the FIOS system, because different data will be combined and can be evaluated in one system. The data model for the integrated data is the introduced RDF Data Cube Vocabulary. By automating the ETL process, the usability increases and thus the user can concentrate on analyzing data.

4.2.1 Architecture Diagram of FIOS 2.0

The FIOS 2.0 system is set up on the FIOS 1.0 system. The system is separated in two logical parts: The ETL Part, which contains the ETL process and the Runtime environment to display and analyze the data. As seen in figure 8 the bases of the data are the data sources: SEC XBRL, provided by the Edgar Wrapper, Yahoo! Finance Data, provided by the Yahoo! Finance Wrapper and other freely available data sources, which are linked to the resources. The first step is to crawl the data from different data sources. (1) Therefore we use LDSpider. While the automation process we included a tool for automated creating the seed lists for crawling data. However, in this system architecture it is summarized into Crawling Data. After Crawling the LDSpider publishes them in a RDF file. This file will be refined while the process. It will be first normalized before a reasoner merges information from equal resources to one. Detailed information about the automation of the ETL process will be explained in section 4.2.2 - Automation of ETL Process. Finally the data will be uploaded into an Open Virtuoso server (2). The Open Virtuoso server has an endpoint, which can be accessed by different applications to present the data. The data in the Open Virtuoso can be accessed via the SPARQL Endpoint (3). Multiple clients can access this endpoint to retrieve data, provided by the Open Virtuoso Server. The language for retrieving data from the Open Virtuoso is SPARQL.

Overview of the FIOS 2.0 system architecture:

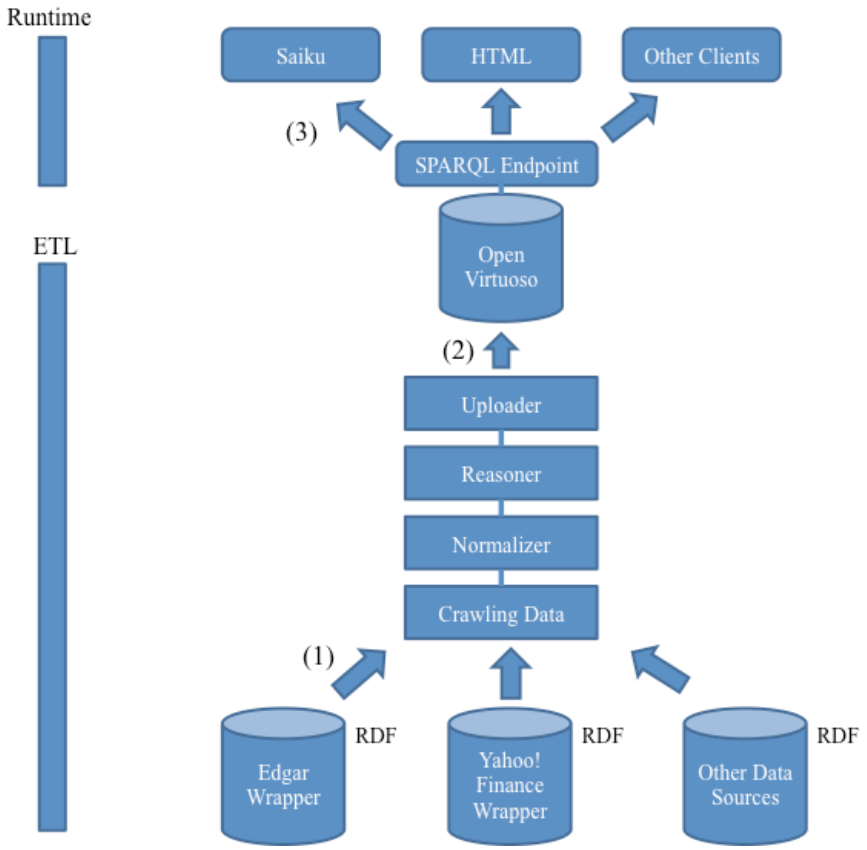


Fig. 8: FIOS 2.0 system architecture

4.2.2 Automation of ETL Process

The automation of the ETL process is the automated execution of the single steps. The automation increases the usability of the FIOS system. The business analysts can concentrate on analyzing the data due to the automated ETL process. Furthermore, the users do not need profound knowledge about the used data model or the data sources. At some parts the user can engage the system and can e.g. add additional seed URIs, which will be explained later, or change the base URI of the normalization step. But if not, the system still works and provides data for business analysts. A representation of the ETL process is given in figure 9.

ETL Process of the FIOS system:

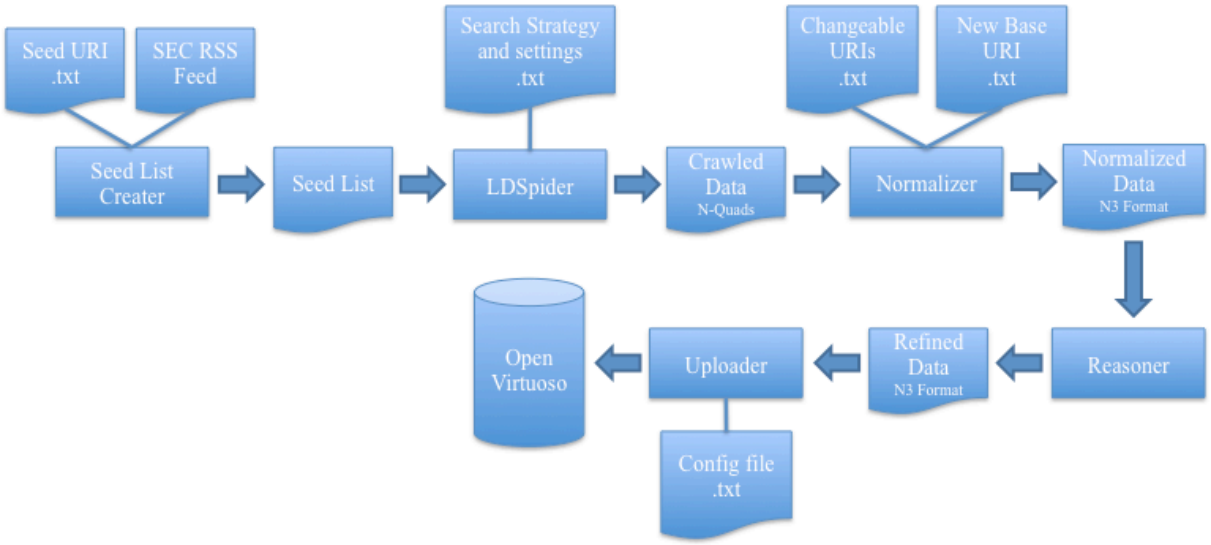


Fig.9: ETL Process of the FIOS system

The process starts at the SeedList Creater. It accesses the RSS feed of the SEC database to check if new SEC filings were uploaded to the SEC database. Furthermore, for already crawled companies, he also generates URIs that lead to the Yahoo! Finance Wrapper for the actual date to keep the stock market data in the data warehouse actual for them. Also the user has the possibility to extend the seed list for his own URIs, which serves as an additional data source. All these URIs will be written into a text file, called seed list. Thus the seed list for crawling data contains now the URIs of the XBRL archives to access the Edgar Wrapper, the Yahoo! Finance archives and potential entered URIs by the user. The seed list will be passed to the LDSpider, a WebCrawler that automatically extracts information from a document and jumps forward to another linked document. The LDSpider uses the URIs in this seed list as access points to start crawling for data. At the end, the entered URIs in the text files will be deleted so the LDSpider will not crawl them again and thus the user does not need to delete the entered URIs himself. The crawled data will be passed to the normalizer. The normalizer receives on the one hand the triple file from the LDSpider as an input and on the other hand two text files. One of the text files contains a list of URIs that should be changed in the triple file by the normalizer. Both, predicates and objects are changeable. The other text file contains the base URI to which the determined URIs should be changed. Therefore, the user can determine which URIs should be normalized by entering the URIs into the corresponding text file. The result is a normalized triple file, which will be passed to the reasoning step. The reasoner now checks if the file contains synonym resources that can be merged by dissolving the owl:sameAs statements. The result of this step is a refined data in a triple file, which will now be uploaded into the data warehouse. This happens by using the bulk load of the Open Virtuoso server. For uploading the data file, the uploader got a text file that contains information about the graph where the triples should be stored. The

detail processes of the single steps are explained in the corresponding chapters. At the end, the uploaded data can be visualized within different tools. The different possibilities to view the data will be introduced in chapter 4.2.2.6 - Visualization. The automation process contains two options. The first is the delta mode. Delta mode describes the daily upload to the database of the crawled data without any engages by users. So, only new fillings will be uploaded. The second mode is the full upload. This mode uploads all data of the RSS feed of the SEC filling and not only the latest fillings. Besides this, for every mode it is possible to define own URIs, which should be crawled an uploaded to the database, by entering the URIs into a text file.

4.2.2.1 Yahoo! Finance Wrapper

We decided in the chapter 4.1.2 - Selection of Data Sources to connect Yahoo! Finance as another data source for the FIOS system. For connecting Yahoo Finance data with XBRL data, a wrapper was needed due to the fact that the Yahoo! Finance APIs do not provide the stock market data in a Linked Data principal. Wrappers hide the complexity from users by automatically transforming an input to the output format. The Yahoo! Finance API publishes the stock market data in a CSV format. Therefore, the wrapper has to adapt this format into a semantic web format. There were two separate Yahoo! Finance APIs. One is for historical stock data and one for actual stock data. Due to the fact that we want to provide both, historical stock market data and actual stock market data, we had to connect both of the APIs. It should be noted that Yahoo! Finance publishes NASDAQ[25] stock quotes. NASDAQ is the biggest electronic stock quotes market in the USA. It is located in New York City. The trading hours are from 09:30 AM to 4 PM EDT[26]. The Yahoo! Finance Wrapper publishes actual stock information at the trading hours. Time differences to other time zones, e.g. CET, should be noted.

The Yahoo! Finance Wrapper has to publish the data according to the data model, presented in chapter 4.1.3 - Data Model. Publishing the data with this data model has two advantages: First, it enables us to keep the data model conform to the RDF Data Cube Vocabulary and to establish the possibility to execute OLAP functions on the data. The other advantage is that it keeps the data model conform to the Edgar Wrapper, which has the same data model. There are different serializations available for publishing Linked Data. One is in triples, where each data is represented in triples. A triple consists of a subject, predicate and object. This is a very simple serialization of the data, which can read by humans well. Another representation format is similar to XML, called RDF/XML, where the information is embedded in tags and attributes. We decided to publish the data in RDF/XML, due to the similar representation of XML, which is a W3C recommendation.

[25] National Association of Securities Dealers Automated Quotations
[26] Eastern Daylight Time

The wrapper is programmed in Java. Java is an object-oriented language that has the advantage of encapsulating parts of the functions and reusing them in other programs. [Flan03] For publishing and hosting the wrapper for everyone in the web we use the Google App Engine[27]. Google provides the infrastructure and resources for easily publishing and managing its applications in the web. However, a restriction to use the Google App Engine exists. Users are only allowed to download 1 GB of data from the App Engine per day. For better managing and creating RDF/XML documents, the JDOM library is used. JDOM[28] is a programming API to easily create and edit XML formatted files in java. For the Yahoo! Finance Wrapper we used the latest available version jdom 2.04.

A UML[29] class diagram of the Yahoo! Finance Wrapper can be seen in figure 10. UML is a language for visualizing circumstances and is probably one of the most used modeling languages for computer systems. [KaBü08] UML comprises multiple languages, among others the UML class diagram. A class diagram shows the classes of a program, its methods and attributes and the relationships between the classes.

Class diagram of the Yahoo! Finance Wrapper:

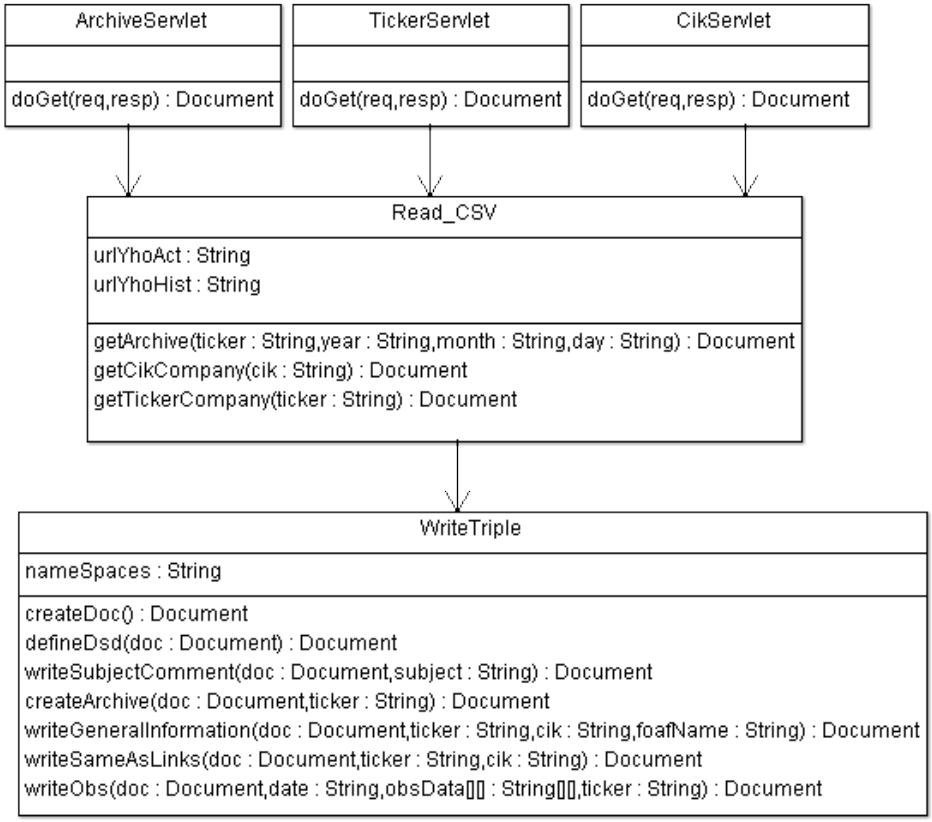


Fig. 10: Yahoo! Finance Wrapper class diagram

[27] https://developers.google.com/appengine/, 7th May 2013
[28] http://www.jdom.org/, 7th May 2013
[29] Unified Modeling Language

If a client access the Yahoo! Finance Wrapper, the Wrapper checks the URL for the keywords “ticker”, “cik” and “archive”. Depending on the called URL, the particular class will be used to handle the URL call and return the corresponding answer. By entering the ticker symbol of a company, the Ticker servlet is used, and by entering a CIK, the CIK servlet is used. A ticker symbol is a unique acronym for a stock market listed company. Ticker symbols contain between one and four letters. The Ticker servlet calls the archive method of the Read_csv class and attaches the ticker to this method. The Read_csv class manages the administration and the call of the corresponding methods in the WriteTriples class. The individual methods will be called up in an order, which depends on the servlet. As a result the class returns a document to the servlet, which will be returned to the user. With the help of the diagram, probably one can see that the document is once generated at the beginning in the method “createDoc()” and then this document will be passed according to a fixed sequence through the different methods within the WriteTriple class, managed by the Read_CSV class mentioned above. The WriteTriple class contains more than only one attribute. However, for the sake of clarity, in this UML class diagram the namespaces were grouped into one. The returned document is a RDF file that lists among others links to archives that contain information of the stock market data for a special date, owl:sameAs links to other data sources that are related to the entered company ticker and some general information about the company e.g. CIK and company name. Like mentioned in the chapter 4.1.3 - Data model, the Yahoo! Finance Wrapper uses for presenting the data existing projects and schemes where it makes sense. For presenting the name of a company we used FOAF metadata.

If users want to get information about the stock market data for this company of a certain date, they have to follow them into the corresponding archives. An archive can be called up according to this syntax:

http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11

The first part, after the “http://yahoofinancewrap.appspot.com/” part, defines that the following String is for calling up an archive, where observations for a special date are located. After “archive/”, the ticker symbol for the requested company follows. After the ticker symbol, the requested date follows. The format for the date is YYYY-MM-DD. The exemplary URL above means that the requester will therefore have all available information about the company MA on the 11th of February 2013.

Due to the keyword “archive”, the wrapper calls up the archive servlet that splits the information in the ticker symbol and the date. Due to the fact that Yahoo! Finance has two APIs, one for historical stock market data and one for actual market data, the wrapper checks if the date is actual or not. If the date is not actual, which means that the date lies in the past for more than one day, the historical API is used. The historical

stock market API is a URI that asks for the ticker symbol and a time interval for which stock market data should be passed. Due to the fact that the Yahoo! Finance Wrapper publishes stock market data in the archives only for one day, the time interval is set in such a way that the beginning time is the same as the ending time. The default URL for calling up the Yahoo Finance historical stock quotes API looks like followed:

http://ichart.yahoo.com/table.csv?s=TICKER

The Wrapper takes this default URL and adds the corresponding ticker symbol at the end. Then the time interval follows:

&a=1&b=11&c=2013 &d=1&e=11&f=2013&g=d&ignore=.csv

The letters a, b and c determine the beginning of the time interval where “a” indicates the month minus one, “b” the date and “c” the year. The letters “d”, “e” and “f” indicate the end of the time interval. The wrapper takes the entered date from the Yahoo! Finance Wrapper API and transforms it to the corresponding format. Finally, the last string &g=d&ignore=.csv is added. The wrapper receives a CSV file with the corresponding stock quotes for the entered company for this date. The CSV file contains two rows. The first row contains the description for the key figures. The second contains the values. The wrapper takes the description of the values and the values and stores them temporarily in a two dimensional array. Each entry in the array corresponds to an observation. The wrapper writes these observations in the method writeObs. Additionally comments and labels for the corresponding subjects are attached. By temporarily saving the subjects and values in a two dimensional array, it is ensured that even if the order of the subjects in the API are changed, the wrapper lists the values and subjects correctly. The presentation of the data follows the RDF Data Cube Vocabulary data model. In the first part of the document is the data structure definition that describes the used measures and dimensions. The measure is the observation value. The dimensions are the issuer for which the information is considered, the date on which the observation has been made, a subject, which indicates the topic of the observation and the segment information. Then the single observations made on this day are listed.

Below, an example of a published observation for the MasterCard Company by the Yahoo! Finance Wrapper is shown. The observation contains among others the information that the open value for a MasterCard stock on the 11th of February 2013 was $525.62.

Observation made on the 11th February 2013 for MasterCard, represented by the Yahoo! Finance Wrapper:

```
<rdf:Description rdf:about="#ds">
    <rdfs:seeAlso>
        <qb:Observation>
            <qb:dataSet rdf:resource="#ds"/>
            <yhof:subject
                rdf:resource="http://yahoofinancewrap.appspot.com/vocab/stock#Open"/>
            <sdmx-measure:obsValue
                rdf:datatype="http://www.w3.org/2001/XMLSchema#double">525.62</sdmx-
                measure:obsValue>
            <dcterms:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2013-
                02-11</dcterms:date>
            <yhof:issuer resource="../../ticker/MA#id"/>
            <yhof:segment>001141391 2013-02-11</yhof:segment>
        </qb:Observation>
    </rdfs:seeAlso>
</rdf:Description>
```

Fig. 11: MasterCard Example of an Observation by the Yahoo! Finance Wrapper

Yahoo! Finance publishes a lot of key figures in its API. However, the Yahoo! Finance Wrapper does not use every key figure. We concentrated on the most important key figures. Adding additional key figures into the wrapper is indeed easy. Extending the URL, which is called by the Yahoo! Finance Wrapper, can include additional key figures. Combination of letters and numbers at the end of the called URL by the wrapper determines which key figures will be extracted from the API. The combinations of letters and numbers needed to add additional key figures to the wrapper can be read on the Internet.[30] A list of available key figures in the wrapper can be seen in table 2.

Data Source	**Extracted Information**
Yahoo! Finance (Historical):	Open Value Close Value Adjusted Close Value Highest share value at a specific date Lowest share value at a specific date Volume
Yahoo! Finance (Actual):	Open Value Price Earnings Ratio Dividend in percent Dividend per Share Market Capitalization 1 Year Estimated Share Value Current Earnings per Share

[30] http://www.jarloo.com/yahoo_finance/, 7th May 2013

	Earnings per Share Estimated for actual Year Earnings per Share Estimated for next Quarter Earnings per Share Estimated for next Year Price/Earnings to growth Ratio (PEG Ratio) 52 week Range of the share value

Table 2: Published key figures of the Yahoo! Finance Wrapper

Due to the fact of using two APIs, one for actual stock quotes and one for historical, the archives do not look the same all the time. If the called up date corresponds to the actual, the actual stock quotes are called, if the called up date corresponds to a former date, the historical stock quotes are called. The historical stock quotes contain the highest and lowest stock quote of this date, the number of shared stock quotes (volumes), the opening and closing stock value of this date and the adjusted closing value. Thus these six key figures are published historically. Calling the wrapper for an actual date, additionally to the mentioned key figure more key figures like the market capitalization, earnings per share and price earnings ratio, is published. Due to the fact that these key figures are not stored in the Yahoo! Finance database, we have to access different APIs to make them available. If we would not access these two APIs we would only have closing value, open value, adjusted closing value, volume, highest and lowest stock value. Due to different available key figures for different date, there are inconsistencies in the archives. Calling up an actual archive, all mentioned key figures in table 2 are listed. Calling up the same archive one day later, only the first six key figures are available, because Yahoo! Finance does not store this information.

4.2.2.2 Crawling with LDSpider

A WebCrawler, also called spider, is a program that automatically collects, browse and extracts information of a certain format. Especially crawlers are used in search applications. A WebCrawler gets a set of URLs and visits those URLs, where it extracts the information. Afterwards the discovered URL links will be stored in the list of URLs. Then the WebCrawler visits the next page in the list. This procedure will be repeated as long as the list is not empty or a stop criterion is fulfilled. [Harb08]

The LDSpider is a WebCrawler that crawls for linked data information in the web. It comprises two different kinds of crawling strategies: Breadth-first crawling and Depth-first crawling. [IUBH13]

Breadth-first crawling is an algorithm method from the breadth-first search. It can be explained more illustratively on a graph where each URL represents one node. Each node can have multiple edges to other nodes. While breadth-first, the crawler starts at a certain node, got from the seed list, and stores each new URL by following the edges going out from the node in an ordered list. The information in the URL will be extracted. Then the crawler goes on with the next node in the list and also stores each

new URL found by following the edges in the ordered list and extracts afterwards the information from the URL. The algorithm repeats this as long as the list is not empty or a stop criterion is fulfilled. In the case of the LDSpider, stop criterion may be the number of visited URLs. The depth-first crawling strategy is also a search algorithm in the graph theory. However this method goes first into the deep. It gets the first URL and extracts all the information found. Then it follows the first edge to a new node and extracts the information on the URL. Afterwards it also follows the first edge to a new node. If there is no edge to a new node the crawler goes back to the last node, which has an edge to a new node that had not been visited by the crawler before. This process will also be done as long as the stop criterion is not fulfilled. More information regarding the two search strategies can be found in Algorithmen - eine Einführung [Corm10].

The Yahoo! Finance Wrapper contains owl:sameAs statements to the Edgar Wrapper, so links between the companies of these two wrappers are secured and the crawler can get to them. The figure 12 shows links between the databases and wrappers to which the LDSpider can get. Links to DBpedia from the Edgar Wrapper is not secured for every company. Due to the interest of clarity, only the biggest data sources, which provide a huge number of data and are also well known, are presented in this figure. There could be more links between other data sources, not mentioned in the figure. Also links from the New York Times API, yago and freebase to other data source are not displayed in interest of clarity.

Connections between data sources, which the FIOS system can use:

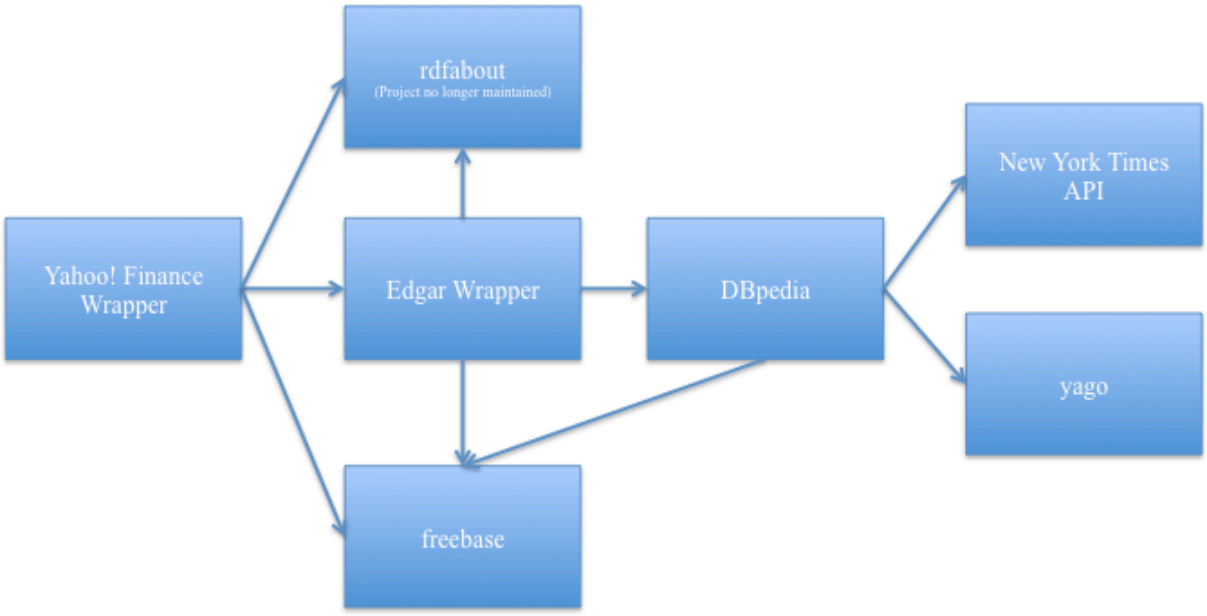


Fig. 12: Visualizing interesting links between data sources

By providing the LDSpider with different points of access, it can follow links of other data sources and crawl there for data. It could be possible that the LDSpider follows the links and crawls data from these data sources that are displayed in figure 12. This depends on the maintained links of the data sources of the resources.

The user can determine the strategy and the stop criterion, used by the LDSpider in this ETL process, by entering the according information in a text file. The Crawling process detects the input made by the user and changes its settings according to the preferences entered in the text file. The stop criterion is the maximum number of crawled URIs. If the user does not change the text file according to his preferences the default settings will be used. Those are: Breadth-first strategy with a maximum number of 200 URIs.

The LDSpider crawls for the information in the web and publishes it in N-Quads. An N-Quad is a serialization for presenting RDF. It contains subject, predicate, object and context information. The context information is the URL where the triple was crawled. This has the advantage of knowing the data source where the information had been extracted. The MasterCard observation known from the previous chapter serves as exemplary crawled data.

Crawled MasterCard observation by the LDSpider:

```
_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/linked-data/cube#Observation>
<http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11> .

<http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11#ds> <http://www.w3.org/2000/01/rdf-
schema#seeAlso>
_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11> .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://purl.org/linked-data/cube#dataSet> <http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11#ds>
<http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11> .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://yahoofinancewrap.appspot.com/vocab/yahoo#subject>
<http://yahoofinancewrap.appspot.com/vocab/stock#Open> <http://yahoofinancewrap.appspot.com/archive/MA/2013-
02-11> .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://purl.org/linked-data/sdmx/2009/measure#obsValue>
"525.62"^^<http://www.w3.org/2001/XMLSchema#double>
<http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11> .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://purl.org/dc/terms/date> "2013-02-11"^^<http://www.w3.org/2001/XMLSchema#date>
<http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11> .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://yahoofinancewrap.appspot.com/vocab/yahoo#issuer> <http://yahoofinancewrap.appspot.com/ticker/MA#id>
<http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11> .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://yahoofinancewrap.appspot.com/vocab/yahoo#segment> "001141391 2013-02-11"
<http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11> .

<http://yahoofinancewrap.appspot.com/vocab/stock#Open> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2004/02/skos/core#Concept> <http://yahoofinancewrap.appspot.com/archive/MA/2013-02-11> .
```

Fig. 13: Exemplary crawled MasterCard data

The LDSpider crawled the observation and prints it in N-Quad. The crawled data also contains the data types in which Yahoo! Finance Wrapper published them. Due to the fact that the observations are not nearly defined, the crawler uses blank nodes to represent the facts. Blank nodes are comprised of "_:" and a random number of letters and numbers. Blank nodes within a data file are unambiguously

The context information determines the data source, where the statement was crawled. However, the Open Virtuoso data warehouse do not allows the uploading of N-Quads, but triple files. Due to this, a transformation step was needed to transform N-Quads into triples. This can be done easily by deleting the context information. This transformation process will be handled in the next chapter during the normalization step of the URIs that simplifies different base URIs into one.

4.2.2.3 Normalization of URIs

The information crawled by the LDSpider is available in N-Quad format. The triples contain the URIs of the corresponding data source. If this data is uploaded into the data warehouse and requests are made on them, the exact URI and also the data model have to be known. However, a goal was to simplify the requests by easing the data model and the URIs. Business analysts do not need to know the underlying URIs to do complex analyzes. By using a normalization step, complex ontologies can be decomposed. This decomposition can reduce the complexity of ontologies and make them simpler. [cf. Rect13] Furthermore, we intend to allow people to browse through the crawled information. However, since this crawled information contains URIs from external servers, people would not be able to browse internal as they will be passed on to external servers. With normalized URIs, the people stay inside of the FIOS system. Therefore, after crawling the information from different data sources, a normalization step had to be included. This step includes the changing of the base URI of each possible data source. The advantage of the normalization process is that the data is consistence to one unique base URI. While changing the base URI, a number that represents the data source from where the data is from, will be added to the base URI. However, the number that will be added is not unique and can change by using different orders in the text file, which determines the changeable URIs. The figure 14 shows an example of normalizing an URI. This figure describes the replacement of the dbpedia.org/resource part into example.org. The number 1 indicates that corresponding number of the data source. By using a number, we can indicate that the corresponding information is from different data sources, even if the resources from different data sources have the same name. If we do not use the number, we cannot indicate information from different data sources.

Normalization of a DBpedia URL using the Normalizer with the new base URI example.org:

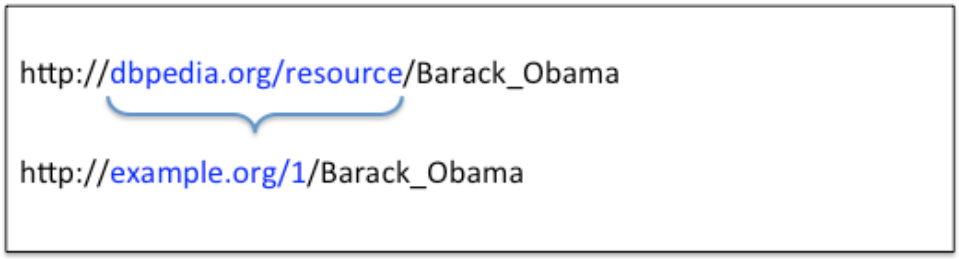


Fig. 14: Exemplary normalization of an URI

The implementation of the normalization process took place in Java. The code is combined in one class. The class takes the crawled information in the n-quad file and first transforms it by removing the context information into a triple file. This step has to be done because the used Open Virtuoso Data Warehouse only allows uploading RDF/XML, Turtle or N3 serializations. While removing the context information, the

class checks for each line if a normalization step has to be done. Therefore, two text files exist. One file contains the new base URI, while the other file contains URIs that should be changed into the new base URI. Changeable in the triple file are both, objects and predicates. By using the text files, which contains the input, the user can choose which URIs should be changed. This increases the possibilities to reuse the program, and the program is also more generic, compared to including the changeable URIs in the code of the program. If the user does not want to change the base URIs, he has to leave the text file empty. Furthermore, the program can normalize different representation serializations of RDF files. Formats like Triple, Turtle or N-Quads can be normalized. Besides, the Turtle serialization is a short type of the triple format and the N-Quads is a triple format with an additionally context information. As the result, the normalizer returns the document in triple form with the corresponding base URI. Besides objects, also properties can be changed using the Normalizer. Therefore the corresponding properties will be inserted into the text file. The approach is therefore the same as normalizing the objects. Both, subjects and properties can be inserted in the text file and the program normalizes them according to the maintained base URI. As result, the normalizer returns the document in triple form with the corresponding base URI.

Our well-known observation serves as an example to illustrate the normalization step. Therefore we determined the following settings in the two text files:

- Changeable URI: yahoofinancewrap.appspot.com
- New Base URI: example.org

The result of normalized observation according to the settings from above can be seen in figure 15.

Normalization of the crawled data from figure 13:

```
_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/linked-data/cube#Observation>  .

<http://example.org/archive/MA/2013-02-11#ds> <http://www.w3.org/2000/01/rdf-schema#seeAlso>
_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://purl.org/linked-data/cube#dataSet> <http://example.org/1/archive/MA/2013-02-11#ds>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://example.org/1/vocab/yahoo#subject> <http://example.org/1/vocab/stock#Open>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://purl.org/linked-data/sdmx/2009/measure#obsValue>
"525.62"^^<http://www.w3.org/2001/XMLSchema#double>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://purl.org/dc/terms/date> "2013-02-11"^^<http://www.w3.org/2001/XMLSchema#date>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://example.org/1/vocab/yahoo#issuer> <http://example.org/1/ticker/MA#id>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://example.org/1/vocab/yahoo#segment> "001141391 2013-02-11"  .
```

Fig. 15: Harmonization of URIs

4.2.2.4 Reasoning

Semantic knowledge bases offer the possibility of enhancing information about objects in a set of logical relations that are machine-readable. As introduced in the Introduction chapter, a particular benefit of the semantic web is the self-closing of new knowledge out of the established knowledge in the knowledge base. [HKRS08, P. 11 f.] The base for generating new knowledge is the semantic reasoner. Based on logical derivation rules, new information that is not given explicit in the knowledge base, will be derivate. The derivated knowledge is also called implicit knowledge. By using reasoners more information can be accumulated. Using logical mathematical methods, reasoners can also check if the given knowledge base is consistent or not. Hereby, the reasoner checks whether there is a not-empty set of individuals or not. An empty set probably indicates an inconsistent knowledge base. A well-known algorithm for checking the consistency of a knowledge base is the tableaux-algorithm. The tableaux-algorithm checks the knowledge base according to inconsistencies by negating the knowledge base and then completes the model by successively adding the rules. A description, how the tableaux-algorithm works, can among others be found in the book *Automated Reasoning with Analytic Tableaux and Related Methods*. [Dyck00] By checking the knowledge base for inconsistencies, we can recognize incorrect knowledge bases and thus warn users of incorrect analysis.

An existing reasoner for OWL is KAON. It is a short cut for Karlsruhe Ontology and was developed by the University of Karlsruhe and the informatics research center Karlsruhe in 2002. Another reasoner is RecerPro, which provides services for OWL and RDF data. These reasoners find synonyms for resources, they implicate subclasses and they check the consistency of the data. So these programs offer a lot of functionality, which we do not need in our reasoning step. To preserve the system of unnecessary or inflated functions we decided to develop our own reasoner whose function is to find synonym resources or rather to dissolve the owl:sameAs statements in the document. This enables the merging of information from different namespaces that all mean the same object. If information brought by Yahoo! Finance Wrapper is related to the same company as information brought by Edgar Wrapper, the reasoner indicates this and merges both information to one resource. The implementation of the reasoner was accomplished in Java.

The reasoner gets a triple or n-quad file as input. Then the program reads every row of the file and checks if the row contains the string "> <http://www.w3.org/2002/07/owl#sameAs> <". The angles brackets ("<" and ">") at the beginning and at the end of the string make sure that the owl:sameAs statement is between the subject and object. Indeed, it is possible that the owl:sameAs statement is also the subject in a triple sentence. This would be the case if the owl:sameAs subject is declared as a property or a label which describes the subject in more detail. If a line contains the string mentioned above, the line will be splitted and the subject and objected will be extracted. Besides of reasoning objects, also properties can be merged if owl:sameAs statement are for the properties available. The next step is the adding of the subject and the corresponding object into a Hashtable. Using Hashing with separated chaining will solve collisions of elements, which are mapped to the same entry of the table. Collisions with chaining will be solved by putting elements that will be hashed into the same entry into a linked list. [CLRS03] We build an Hashtable with the help of the java class "HashMap<K,V>", where K is the key to insert the object V. The key maps a String to a corresponding Hashtable entry. However due to the fact that multiple objects will be mapped to the same key, the object V has to be a data structure that can store multiple objects. We decided to use a java HashSet for saving the Elements. A HashSet is data structure for saving a set of elements. HashSets are fast data structures, which do not consider the input sequences. [FS99] The method's input, delete and access will be executed in a constant time. [KS09] Before inserting a new element into the HashSet, the structure itself checks if the element is already in the HashSet, so this verification part must not be implemented by us. Therefore, we use a HashSet within the HashMap structure. The HashMap stores the HashSet while the HashSet stores in turn the subjects and objects of the owl:sameAs statements. In the following, this self-generated structure will be called Hashtable. The Hashtable insert method checks if one or both of the subject and object are already contained in the Hashtable. If so, the corresponding HashSets will

be merged into one. Also the key value to indicate the HashSet will be added to the list, due to the fact that now it is no key anymore, but a part of the HashSet. This approach makes sure that same elements are stored in one HashSet. Due to the HashSet structure, the merging of two HashSets can easily be done via one function, called “addAll”. If both elements are not in the Hashtable, a new entry will be added to the Hashtable by using the subject as the key and a new HashSet, which contains the subject, as entry. Figure 16 shows the merging of two HashSets, if the subject and object are both already in the table. The figure shows linked lists as entries of the HashMap, as it is clearer and more readable. However, we used HashSets and not Linked Lists as the corresponding data structure.

Illustration of the Hashtable while the reasoning process:

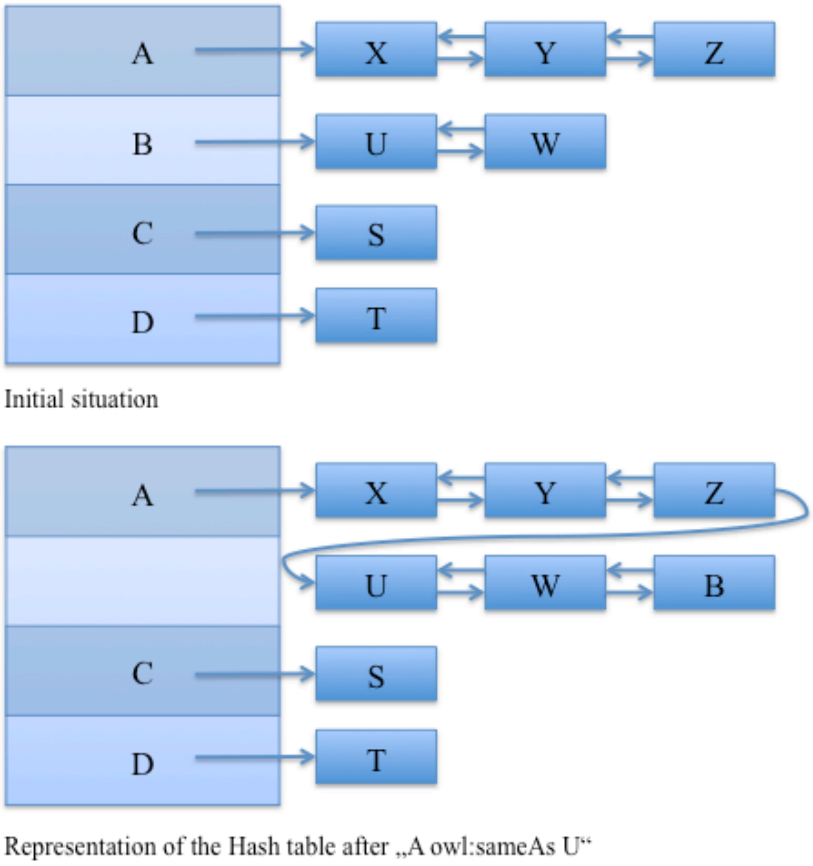


Fig. 16: Merging two HashSets due a owl:sameAs statement

The corresponding key of the Hashtable will replace each element in the RDF file that appears in the linked list. In the example of the figure 16 the elements X, Y, Z and U, W and B will be replaced by A because they will be mapped into one HashSet. The replacement of W and B by A can easily be reconstructed due to the transitivity of the owl:sameAs statement. Because A is the same as U, and U is the same as W, W can also be replaced by A. This logical transitivity can be applied also for the other examples in the linked list in figure 16.

After hashing the values into the Hashtable, the program replaces the subjects, objects or properties with the corresponding entry of the Hashtable for each line. Therefore, the program looks up the name of the resource of the triple file in the Hashtable. The corresponding key of the Hashtable, in which the value appears, replaces the value in the triple file. This process will be done till the end of the triple file. After replacing the subjects and objects, the reasoning step is finished.

To illustrate the approach of the reasoner we have to extend our example from above. Therefore a new triple is included:

```
<http://example.org/2/1141391#id> <owl:sameAs> <http://example.org/1/MA#id> .
```

Due to the owl:sameAs statement, both the number 1141391[31] from data source 2 and the ticker symbol MA from the data source 1 will be mapped into one entry of the Hashtable. This means that same URIs represent the same resource. The reasoner merges now all the information of the resource http://example.org/1/MA#id with the information from the resource http://example.org/2/1141391#id. The result is the following:

Result of the reasoning step with the file from figure 15 and the owl:sameAs statemen above:

```
_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/linked-data/cube#Observation>  .

<http://example.org/archive/MA/2013-02-11#ds> <http://www.w3.org/2000/01/rdf-schema#seeAlso>
_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://purl.org/linked-data/cube#dataSet> <http://example.org/archive/MA/2013-02-11#ds>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://example.org/vocab/yahoo#subject> <http://example.org/vocab/stock#Open>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://purl.org/linked-data/sdmx/2009/measure#obsValue>
"525.62"^^<http://www.w3.org/2001/XMLSchema#double>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://purl.org/dc/terms/date> "2013-02-11"^^<http://www.w3.org/2001/XMLSchema#date>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://example.org/vocab/yahoo#issuer> <http://example.org/cik/1141391#id>  .

_:httpx3Ax2Fx2Fyahoofinancewrapx2Eappspotx2Ecomx2Farchivex2FMAx2F2013x2D02x2D11xxbnode5
<http://example.org/vocab/yahoo#segment> "001141391 2013-02-11"  .
```

Fig. 17: Exemplary MasterCardt triple file after reasoning

[31] This is actually the CIK of the MasterCard Company

This is the end of the transformation process of our observation. The last step is to upload it to the Open Virtuoso server. This is explained in the next section.

4.2.2.5 Storage with Open Virtouso

Open Virtuoso is described as an universal server because it offers multiple server-side protocols. Among others, it offers Database-server functionality for storing data[32]. Data that can be stored in the Open Virtuoso server ,which includes RDF data. The RDF data will be stored as graphs in the database. A graph database provides a graph abstraction to the stored data. [YMYM10]

Open Virtuoso offers the possibility of Bulk Loading RDF source files into the database. Bulk Load is a method to load a high number of data efficiently from flat files into the database. This function is used to upload RDF data into the Open Virtuoso database. To enable the uploading of data within the automated ETL process, the "upload_setting" text file must be maintained. Within this text file the graph in which the data should be uploaded must be maintained. Without this setting, the uploading process will not start. According to the settings specified in the text file the data is uploaded into the Open Virtuoso and ready for use. The different possibilities to view the data are part of the next chapter.

4.2.2.6 Visualization

The open virtuoso server provides a SPARQL endpoint, which clients can access. Multiple programs can use this endpoint, which provides a variety of applications to use this endpoint. Every analyst has his own preferred representation of data. To allow every user to access the data according to his preferences, the FIOS system offers different possibilities to represent the data. The first representation possibility is a HTML page on which the data is displayed. Therefore we use SPARK, a JavaScript library to display the SPARQL requests in different formats on the HTML page. Different statistics and share prices are displayed, up to balance sheets data, which compare the share performance with the balance sheets. In addition to that information, also meta information, e.g. number of employees and a short abstract of the company, is mentioned if available in the data warehouse. Besides the HTML page, Pubby can be used to explore the data in the data warehouse. This tool provides a linked data interface for SPARQL endpoints and displays the resources and the according triples in the web browser. The last visualization option is Saiku, an open-source analytical tool, which offers OLAP. With this tool, the RDF Data Cube Vocabulary can be used to analyze the data with OLAP functions.

[32] http://www.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtuosoFAQ#Why is Virtuoso described as a Universal Server?, 7th May 2013

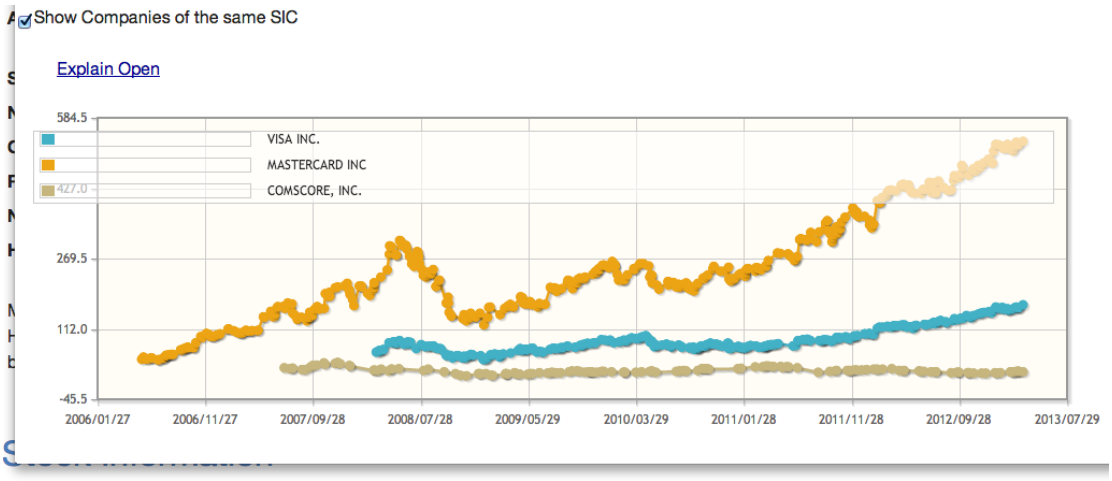
4.2.2.6.1 HTML

To display the data in the data warehouse on a HTML[33] page, we use SPARK[34]. SPARK is a JavaScript library that allows accessing the SPARQL endpoint and displays the query results on a HTML page. SPARK requires jQuery to run and displays the results given from the endpoint. jQuery is also a JavaScript library, which makes the development of programs and tools much simpler, especially in the AJAX sector. SPARK and jQuery can be used with all available browser. SPARK retrieves a request that is coded in the HTML page, sends this to the SPARQL endpoint of the data warehouse and receives the corresponding result to the query and displays it on the HTML page. The data can be displayed in both, tables and charts. There are two different charts available: Pie chart and date chart. The date chart is appropriate e.g. to display the process of the stock value over time. The pie chart is suitable e.g. for showing the ratio of different key figures between companies in the same business sector.

As an example for representing finance data with SPARK on a HTML page serves our MasterCard data, which we crawled, normalized, reasoned and uploaded it to Open Virtuoso. Figure 18 shows a section of the HTML page. In this picture, the progress of the open stock value of the MasterCard compared to companies within the same SIC can be seen and also the actual values for the different key figures, available in Open Virtuoso. Our observation crawled from the Yahoo! Finance Wrapper is part of the date chart, which shows the progress of the open stock value. The MasterCard curve is the one on the top, followed by VISA Inc. and Comscore Inc..

[33] Hypertext Markup Language
[34] http://km.aifb.kit.edu/sites/spark/, 7th May 2013

Visualization of the MasterCard example on the HTML page with SPARK:



Subject	Date	Value	Avg 2013	Avg 20
Open	2013-03-27,	532.21,	520.9531578947367,	430.62
High	2013-03-27,	540.73,	525.1484210526315,	434.07
Low	2013-03-27,	530.38,	517.6799999999999,	426.12
Close	2013-03-27,	538.03,	521.6921052631579,	430.40
Volume	2013-03-27,	648300,	709942.1052631579,	798382
AdjClose	2013-03-27,	538.03,	521.6763157894738,	429.75
PriceEarningsRatio	2013-03-27,	24.52,	24.33846153846154,	
DividendYield	2013-03-27,	0.22,	0.23,	
EarningsPerShare	2013-03-27,	21.94,	21.56607692307692,	

Fig. 18: Example of the possibilities with SPARK

4.2.2.6.2 Pubby

Pubby is a frontend tool, developed by the Freie Universität zu Berlin. The tool accesses the SPARQL endpoint provided by the data warehouse and displays the Linked Data in triples. Pubby handles requests to the mapped URIs according to the mapping configuration. The resources that map the mapping file will be displayed in the browser and not redirected to the original URI. Pubby runs within a Tomcat servlet container. The design and technical structure can be seen in figure 19.

Technical design and structure of Pubby:

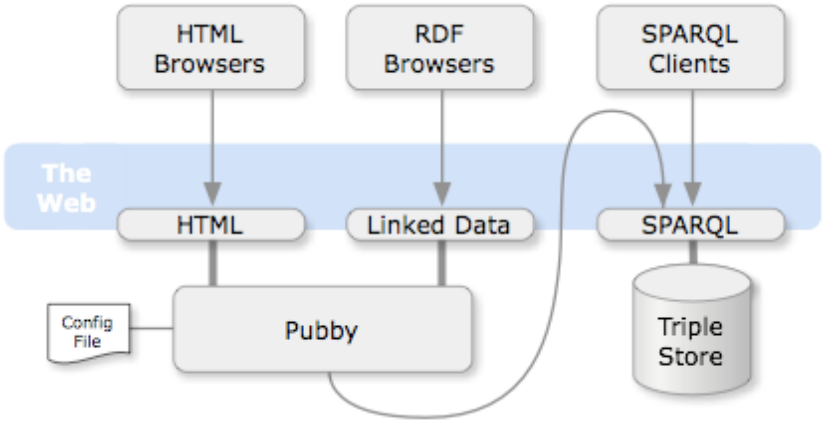


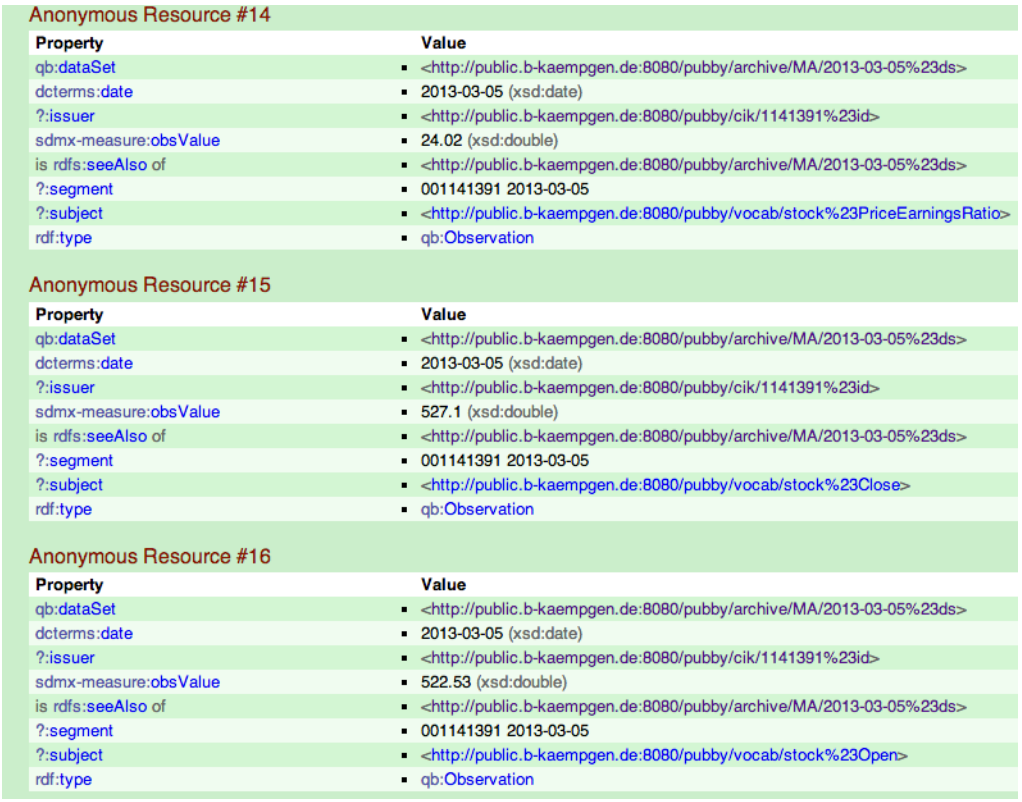
Fig. 19: Technical design of Pubby[35]

Before Pubby can be used, the configuration file have to be maintained. In this file, the prefixes are maintained and the SPARQL endpoint with the dataset base for the resources that will be displayed. Other resources that do not fit to that base URI will not be displayed in Pubby but passed to external servers. Afterwards, Pubby is ready for use and displays all resources that begin with the maintained base URI. The worth of Pubby is that it enables the user to discover the data by him. The user can see which data is in the data warehouse and can get a deeper insight into the data model. The user additionally sees data that is possibly not displayed in Saiku or the HTML site due to the fact that the information is not part of the observations in the data. Moreover, before users want to access the endpoint with their own program, they can first discover the data model and see if their program fits to the data model.

A figure of our well-known exemplary observation can be seen below. In this image, besides our crawled data, also other observations of the same archive can be seen. The Anonymous Resource #16 is the considered observation from the previous chapter.

[35] http://wifo5-03.informatik.uni-mannheim.de/pubby/, 7th May 2013

Representation of data in Pubby:



Anonymous Resource #14

Property	Value
qb:dataSet	<http://public.b-kaempgen.de:8080/pubby/archive/MA/2013-03-05%23ds>
dcterms:date	2013-03-05 (xsd:date)
?:issuer	<http://public.b-kaempgen.de:8080/pubby/cik/1141391%23id>
sdmx-measure:obsValue	24.02 (xsd:double)
is rdfs:seeAlso of	<http://public.b-kaempgen.de:8080/pubby/archive/MA/2013-03-05%23ds>
?:segment	001141391 2013-03-05
?:subject	<http://public.b-kaempgen.de:8080/pubby/vocab/stock%23PriceEarningsRatio>
rdf:type	qb:Observation

Anonymous Resource #15

Property	Value
qb:dataSet	<http://public.b-kaempgen.de:8080/pubby/archive/MA/2013-03-05%23ds>
dcterms:date	2013-03-05 (xsd:date)
?:issuer	<http://public.b-kaempgen.de:8080/pubby/cik/1141391%23id>
sdmx-measure:obsValue	527.1 (xsd:double)
is rdfs:seeAlso of	<http://public.b-kaempgen.de:8080/pubby/archive/MA/2013-03-05%23ds>
?:segment	001141391 2013-03-05
?:subject	<http://public.b-kaempgen.de:8080/pubby/vocab/stock%23Close>
rdf:type	qb:Observation

Anonymous Resource #16

Property	Value
qb:dataSet	<http://public.b-kaempgen.de:8080/pubby/archive/MA/2013-03-05%23ds>
dcterms:date	2013-03-05 (xsd:date)
?:issuer	<http://public.b-kaempgen.de:8080/pubby/cik/1141391%23id>
sdmx-measure:obsValue	522.53 (xsd:double)
is rdfs:seeAlso of	<http://public.b-kaempgen.de:8080/pubby/archive/MA/2013-03-05%23ds>
?:segment	001141391 2013-03-05
?:subject	<http://public.b-kaempgen.de:8080/pubby/vocab/stock%23Open>
rdf:type	qb:Observation

Fig. 20: Pubby presenting our well-known MasterCard data example

The base URI "public.b-kaempgen.de:8080" in figure 20 is used with Pubby. However, in the database, the data is stored with the base URI known from the examples above. Pubby displays the dataset base prefix according to the server on which it runs. Other URIs, that do not fit to the base URI, maintained in the config file of Pubby, will not be displayed in Pubby, but passed to an external page.

4.2.2.6.3 Saiku

The OLAP analytic tool Saiku offers ad-hoc analyzes with OLAP functions in real-time. Ad-hoc analyzes are reports that can be changed fast and according to the needs of the analysts. OLAP functions, e.g. Slice/Dice, can be used in Saiku. The data can be displayed in tables and also in charts. Different charts are available e.g. Stacked Bars, Line chart, Pie Chart and Heat Grid. The different charts are well balanced and enable the representation of data according to the best view. Moreover, Saiku allows the export of the data in XLS format. The data in the spreadsheet format can be further processed and used for analytical purposes. Due to the export feature of Saiku, the data can also be analyzed in Excel with all functions of the Excel tool. As a result of that, users can also further use Excel by exporting their data from Saiku.

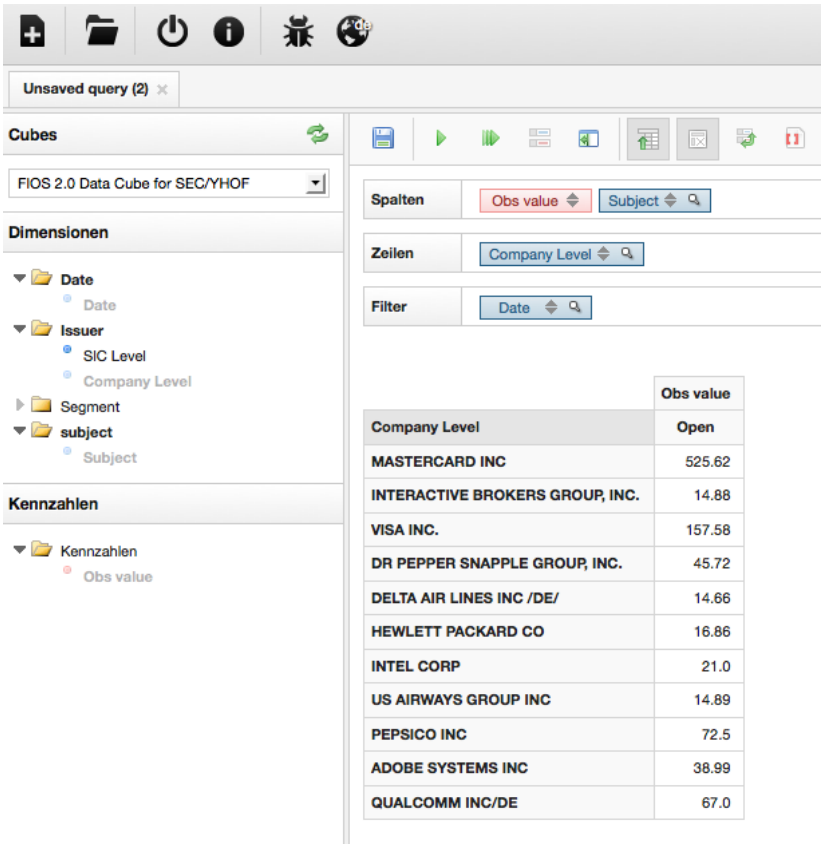
Saiku itself has no feature to access Linked Data sources therefore Olap4ld will be used. Olap4ld is an open-source engine that implements the Open Java API olap4j for accessing triple stores and allows OLAP analyzes. Olap4j is an API for OLAP servers,

such as Open Virtuoso is. Olap4ld translates the OLAP queries into SPARQL and sends it to the triple store endpoint and also transforms the results in such a way that Saiku can display the data. By the use of this open-source engine and the RDF Data Cube Vocabulary, users do not need to know the data model or have to create a SPARQL request. The handling of the OLAP functions is therefore as simple as possible and easy accessible for business analysts.

The Saiku interface is very simple. Users can create reports via drag and drop and thus assign the key figure and measures to the column or rows. Business analysts can analyze share stock values and balance sheets of companies and sectors by using OLAP. Due to the simple interface of Saiku and the well-known OLAP functions, the use and the possibilities for analyzing data are suitable for business analysts.

A representation of Saiku with our observation can be seen in figure 21. The example shows the open stock value for all available companies in our Open Virtuoso Server at the 11th of February 2013.

Representation of our MasterCard example in Saiku compared to other companies at the same Date:



Company Level	Obs value Open
MASTERCARD INC	525.62
INTERACTIVE BROKERS GROUP, INC.	14.88
VISA INC.	157.58
DR PEPPER SNAPPLE GROUP, INC.	45.72
DELTA AIR LINES INC /DE/	14.66
HEWLETT PACKARD CO	16.86
INTEL CORP	21.0
US AIRWAYS GROUP INC	14.89
PEPSICO INC	72.5
ADOBE SYSTEMS INC	38.99
QUALCOMM INC/DE	67.0

Fig. 21: Representation of data in Saiku

4.3 Evaluation: FIOS 2.0 for the XBRL Challenge 2013

This chapter deals with the evaluation of our FIOS 2.0 system. The first analyze of the FIOS 2.0 system evaluates the system according to the criteria of the XBRL Challenge 2013. There we will discuss if the system complies with the criteria of the XBRL Challenge. In section 4.3.2 - FIOS 2.0 we will explain in which environment the FIOS 2.0 system ran for the challenge and how the concrete implementation of the FIOS system looked like. The section 4.3.3 - Requirement Coverage Analysis deals with the target-actual comparison. We will discuss if the connection of a new data source was successful and whether the available data source can comply the business questions and expectations made in the section 4.1 - Requirements For Integrated Financial Analysis. At the end, the knowledge gained while implementing the FIOS 2.0 system will be summarized.

4.3.1 Evaluation Regarding the Criteria of the XBRL Challenge

The XBRL Challenge is a yearly contest proclaimed by the XBRL US. Individuals or teams can submit their entries to the challenge. As prerequisite, the submitted applications have to be based on XBRL data. The judges, who are mainly executives from US public companies, rate the applications and select the winner. The criteria for judging the entries are the following:

- Improves Access
- Usability
- Analytics
- Design

In the following, the FIOS 2.0 system is evaluated according to those criteria.

The criterion **Improve Access** characterizes the access to corporate data. This criterion is fulfilled by the FIOS 2.0 system due to the integrated data from SEC XBRL and Yahoo! Finance. Investors and stakeholders can access data from both data sources within one system. The criterion **Usability** describes the possibilities for stakeholders or investors to access the data. Furthermore, this criterion also describes the quality and usability of the application. Multiple access points to the available data are preferable. Regarding the FIOS 2.0 system, the endpoint of the Open Virtuoso Server provides access to the data. Based on this endpoint the introduced applications can access the data and visualize them in different ways. We have a HTML page to present on a predefined way the data. Then there is also Pubby, an application for browsing through the data provided by the endpoint. As the last access possibility to the FIOS system, Saiku offers OLAP functions to analyze the financial data. Each application has its advantage. Pubby is useful for discovering the data while HTML enables getting daily static reports. Saiku, as the last visualization application, is preferable for

ad-hoc analysis. By using the OLAP functions user can answer their individual business questions. However more tools can be added and access the endpoint. So there is no restriction on the number of tools. The third criterion, the **Analytics**, describes the functions for analyzing the data. At least, the XBRL commission demands year-to-year comparisons, comparisons between different companies or other functions to help investors to make their decisions. The Pubby application is not sufficient for analyzing the data in a suitable way as the commission demands it. Pubby allows only the browsing of the data. However comparisons can be made by browsing through the data and calling the corresponding observations side by side, but as mentioned this is not suitable. More suitable is the HTML page, where the web page contains the latest information and the progress of the key figures. By following the course of the stock market values or balance sheet items, year-to year comparisons are possible. Also the progress of key figures from other companies within the same SIC can be faded in, so that comparisons to other companies are possible. However, the comparison is limited to companies within the same SIC. While creating the HTML page we decided only to allow users to fade in companies of the same SIC, due to the fact that allowing displaying all companies would lead to a confusing representation. The last possibility to analyze the data is the use of Saiku. This tool allows users to execute OLAP functions on the data in the data warehouse. The OLAP functions, combined with the simple user interface of Saiku, allow the creation of easy ad-hoc analyzes. Business analysts can do year-to-year comparisons as well as comparisons between companies. Also the possibility to aggregate the data on a higher level is possible. Therefore, the data will be aggregated within the industry sector. This aggregation is ensured by the SIC of the companies. Thus not only companies can be analyzed, but also industry sectors. The last criterion, the **Design**, characterizes the creativity of the submitted application. Therefore the judges check the application whether the idea of the system is known before or original, so the application can be drawn from existing tools for analyzing financial data. In our opinion the FIOS 2.0 system also fulfills this criterion. The representation of financial data according to the Linked Data principle and using them for analyzing purposes is original. A similar approach had not yet been submitted to the XBRL Challenge. Also the idea of providing both, SEC XBRL and Yahoo! Finance data, had not yet been served by other participants of the challenge. Although Calcbench.com displays actual stock market data for a company, but does not offer the possibility to involve them in the analyzing process as we do.

In conclusion, it is obvious that the FIOS 2.0 system fulfills the criteria of the XBRL challenge. There are some improvement possibilities, e.g. the possibility to use Saiku with olap4ld also with other browsers instead of only with Mozilla Firefox. However, the restriction on a certain browser limits the usability only slightly. The possibility to display more than one dimension in Saiku would be preferable, as now only one dimension can be represented, which limits the number of possible business questions.

4.3.2 FIOS 2.0

This chapter will introduce special settings made on the FIOS 2.0 system for the XBRL Challenge 2.0. For the XBRL challenge we took our development, introduced in chapter 4.2 – Integration of SEC XBRL and Yahoo! Finance Data for Analysis and adapted it to the XBRL Challenge. First, the settings, made to the ETL process according to the XBRL challenge, will be introduced in section 4.3.2.1 – ETL Process. The next section shows visualizations of the results to see the representation of the data. The last chapter deals with an algorithm analysis of the programmed normalizer and reasoner. The algorithm analysis will show us if the programmed normalizer and reasoner are efficient or not.

4.3.2.1 ETL Process

The introduced FIOS 2.0 system was used for the XBRL Challenge 2013. The extraction process was automatically started each day. However, for the ETL process some special settings were done.

Crawling:

For crawling data the program Seed List Creator automatically accesses the RSS feed of the SEC. However, users have the chance to write own URIs, which should be crawled, in a text file. We filled this text file among others with Yahoo! Finance Wrapper URIs and Edgar Wrapper URIs of companies especially of the computer industry. Therefore we crawled e.g. for Apple Inc., Microsoft Corp., Dell Inc., Hewlett Packard Co. and Intel Corp. During the crawling process of the LDSpider, these URIs were included. Also settings regarding the search strategy can be maintained. However, we did not change them because the basic settings with a maximum number of 200 URIs fulfill our requirements to the system regarding the available data completely. As standard crawling strategy the ETL process uses the breadth-first strategy. The Breadth-first strategy is therefore preferable because we want to have a lot of information about a company in different data sources. By using the breadth-first strategy we ensure that the information on a high number of linked data sources will be crawled primarily. If the depth-first strategy would be used, the crawler gets first a lot of information about topics that are related to a URL that is in turn related to the company. This search algorithm is not preferable in this case, because it delivers unfeasible results.

Normalizer:

The normalizer changes base URIs maintained in a special text file into the base URI maintained in another text file. The base URI, into which the URIs should be changed, was "http://public.b-kaempgen.de:8080", which conforms to the URI of the Server

provided by Benedikt Kämpgen in which the FIOS 2.0 system ran. The URIs, which had to be changed into the server URI were the following:

- http://yahoofinancewrap.appspot.com
- http://edgarwrap.ontologycentral.com
- http://dbpedia.org/resource
- http://de.dbpedia.org/resource

These URIs were changed into "http://public.b-kaempgen.de:8080".

Reasoner:
For the reasoning step no settings can be maintained. The user has no opportunity to intervene in the reasoning process. Therefore we did not change anything according to the FIOS 2.0 system.

Uploading:
The RDF data will finally be uploaded to an Open Virtuoso Server. An Open Virtuoso Server was running inside of the Server from Benedikt Kämpgen. We used this Server as the database. We used the graph “http://fios:saiku“ to manage the data. Therefore the corresponding graph was maintained in a text file, which contains the settings for uploading the data for the ETL process. The ETL process executes the corresponding execution statements for uploading the RDF file into Open Virtuoso. Thus the data is available in the database and can be queried by the visualization tools.

In total there were over 4.49 million triples for the XBRL Challenge in the Open Virtuoso Server, of which were 339,818 observations.[36] This shows that a high number of data were crawled for the FIOS system. This data can be used for analyzing companies. Furthermore, we detected 3,886 subjects in the system. The subjects are the name of the key figures, which are needed to know which topic the value in the observation describes. The next section shows the possibilities to display these triples in the data warehouse.

4.3.2.2 Visualization Screencasts

For the XBRL Challenge 2012 we created a video screencast to show live the possibilities of the system.[37] In addition, we want to show in this section the results of the FIOS system in a few screencasts.

We first created a HTML page for one company, as a pattern page, where we realized our ideas of how the page may look like. The first part of the page displays some general information about the company. In this part among others, the name, number

[36] Detected on the 7th of May 2013
[37] http://www.youtube.com/watch?v=zLqsQ-YHMvk, 7th May 2013

of employees, operating income and abstract of the company is displayed. The second part of the page includes the latest stock values and averages for the year 2013 and 2012. By moving the mouse over a key figure, a short description of the key figures appears. SPARK also retrieves these descriptions via a SPARQL request. Below the latest stock values, a date chart appears that shows the approach of the adjusted closing price for the company. Available balance sheets data can be viewed on an extra HTML page for a better overview. The data is displayed in a balance sheet. However, due to the missing template of a balance sheet and the missing information in which sequence the data should be displayed, we designed a balance sheets data by our own. By this, there is no guarantee of completeness of the data. Some data may be missing and thus the representation of the balance sheet is not sufficient. However, the complete representation of the data can be provided in a list, but this representation is confusing. Due to this, we decided to display the main items of the balance sheet data, even if it is not a complete representation of all available data.

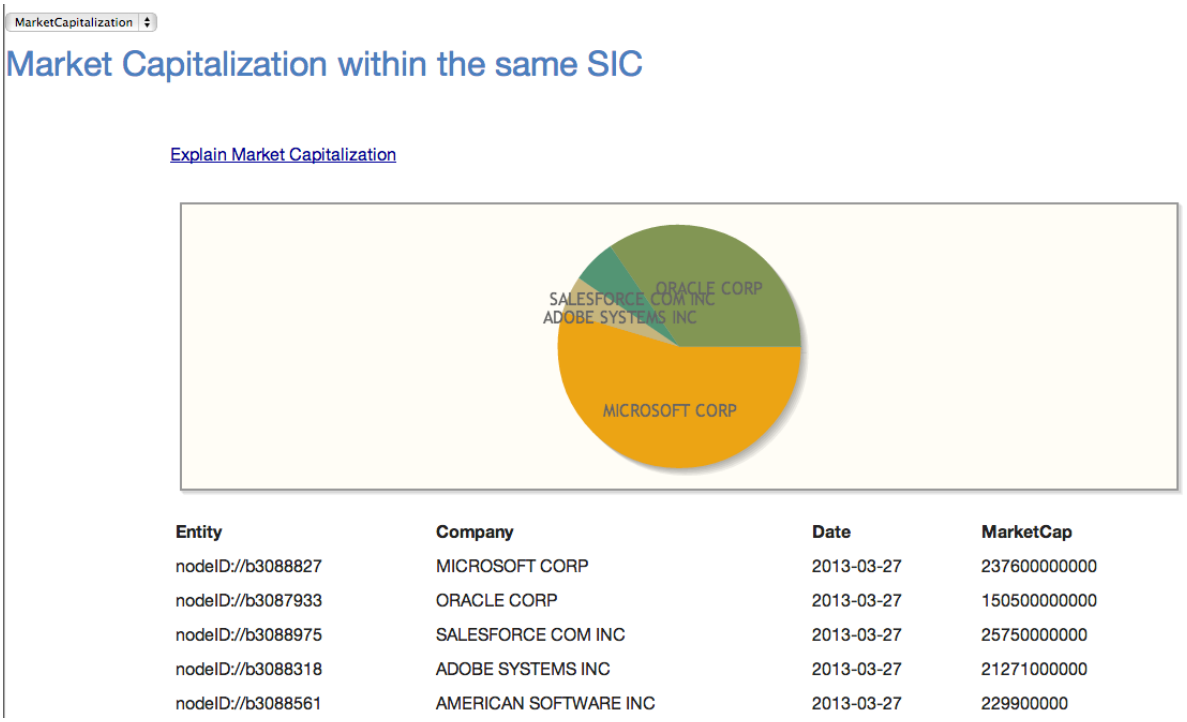
After creating the first page for a company, we decided to use this as a template for all companies. The aim was to type in only the CIK and the users receive the corresponding HTML page where the information for the company is represented. We decided to extend the HTML page and to use PHP[38]. PHP allows the dynamic generation of HTML code. This helped us to create the SPARQL requests according to the CIK the user entered. Depending on the CIK the user enters, the SPARQL requests and also the HTML code will be created. Therefore, users can view information for all companies available in the data warehouse. To help users to find the CIK, we also provided a search interface. Users can search for the name of the company, the SIC and CIK or Ticker. The search function is also executed as a SPARQL request. Within the search mode, regular Expressions are used in the SPARQL requests. Therefore all listed information conforms to the entered input of the user if he uses the search function. A further improvement we did on the HTML page is the possibility to display all key figures, stock value and balance sheet items, in a date chart. Calcbench.com also provides this feature, but in our implementation, users can also enter a filter period, so the date chart only displays the data within this period. This allows zooming and a detailed view of the data to see trends and developments of the data over a certain period. This date chart can be faded in and out via a checkbox. This improvement goes beyond of what Calcbench offers, as users can only see the data of one company over time in a date chart.

Figure 22 is a part of the HTML page for the XBRL Challenge. In this part the companies of the Industry group 7372 “Services-Prepackaged Software“ are visualized according to the market capitalization on the 27th of March 2013. As seen the Microsoft Corp has the biggest part of the market share. The table below the chart

[38] PHP: Hypertext Preprocessor

shows the exact value of the market capitalization. The visualization is made possible by using SPARK, a JavaScript library to visualize SPARQL requests either in tables or in charts.

Example of the HTML page, displaying Microsoft Corp., for the XBRL Challenge 2013:



Entity	Company	Date	MarketCap
nodeID://b3088827	MICROSOFT CORP	2013-03-27	237600000000
nodeID://b3087933	ORACLE CORP	2013-03-27	150500000000
nodeID://b3088975	SALESFORCE COM INC	2013-03-27	25750000000
nodeID://b3088318	ADOBE SYSTEMS INC	2013-03-27	21271000000
nodeID://b3088561	AMERICAN SOFTWARE INC	2013-03-27	229900000

Fig. 22: Part of the HTML implementation for the XBRL Challenge

As mentioned above, also balance sheets data can be seen for companies. The data of the balance sheet is grouped in years. By selecting the corresponding year at the radio bottom, the corresponding balance sheet data appears. However, as noted, there is no guarantee of completeness of the data. There is the possibility of displaying all available data for a certain balance sheets, by creating a list. But this would mix up assets and liabilities items and no structure of the data exists. Thus analyzing or having an overview of the data is complicated. So we decided to create a structured balance sheet, even if data could be missing.

Balance Sheet pattern, displaying Adobe Systems Inc., for the XBRL Challenge 2013:

Balance Sheet for ADOBE SYSTEMS INC,

Select the shown Year

○2012 ◉2011 ○2010 ○2009

Assets

Name	2011-03-04,	2011-06-03,	2011-09-02,	2011-12-02,
Cash and Cash Equivalents	900156000,	827475000,	769212000,	989500000,
Restricted cash-ligitation escrow				4280000000,
Securities Pledged to Creditors				
Corporate Equity Securities				
Corporate Debt Securities				
Mortage- and asset-backend Securities				
Derivatives				
Invesments				
Total Financial Instruments Owned				
Trading				
Prepaid expenses and other current assets				
Investment securities avaiable-for-sale				
Investment in marketable securities	1736679000,	1798045000,	1950064000,	1922192000,
Accounts receivable	533353000,	568570000,	559320000,	634373000,
Materials and Supplies				
Financing receivables				
Inventories				
Restricted security deposits held from customers				
Vendor non-trade receivables				
Prepaid expenses		127211000,	119513000,	133423000,
Other Current Assets				
Deffered income taxes	66928000,	68017000,	71087000,	91963000,
Total Current Assets	3350798000,	3389318000,	3469196000,	3771451000,
Long-term marketable securities				
Equity and Other Investments				
Property, equipment and technology, net	453497000,	463415000,	499059000,	527828000,
Deffered income taxes				
Goodwill	3686073000,	3693505000,	3740199000,	3849217000,
Intangible Assets	447616000,	424199000,	419824000,	545526000,
Other intangible assets, net				
Other assets	164801000,	162040000,	157241000,	89922000,
Total Assets	8310024000,	8339716000,	8492758000,	8991183000,

Liabilities

Name	2011-03-04,	2011-06-03,	2011-09-02,	2011-12-02,
Current maturities of debt and capital leases	1511553000,	1509428000,	1507278000,	1505096000,
Accounts Payable	54742000,	60533000,	65236000,	86660000,
Corporate Equity Securities				
Corporate Debt Securities				
Total Financial Instruments Sold, not yet purchased				
Postretirement Benefits other than Pension				
Accrued compensation and benefits	165912000,	198720000,	182011000,	235500000,
Accrued Taxes				
Income Taxes	57096000,	40970000,	61220000,	42634000,
Accrued Liabilities	458463000,	496535000,	450343000,	554941000,
Deffered Revenue	399572000,	438078000,	439690000,	476402000,
Securities Lending Payable				
Restricted security deposits held for customers				
Brokers, Dealers and Clearing Organizations				
Accrued litigation				
Accrued expenses	458463000,	496535000,	450343000,	554941000,
Other current liabilities	184053000,	172070000,	139916000,	157008000,
Total Liabilities and Equity	985532000,	1050379000,	1032259000,	1250779000,
Long-term Debt				
Long-term unearned revenue	43826000,	43949000,	44369000,	55303000,
Deferred income taxes	120756000,	121996000,	151065000,	181602000,
Other liabilities	44922000,	44323000,	42782000,	50883000,
Total Liabilities	2884617000,	2950301000,	2928800000,	3208070000,
Additional paid-in-capital	2529789000,	2611997000,	2680407000,	2753896000,
Class A treasury stock	3178769000,	3505559000,	3557234000,	3529529000,
Accumulated Income	6045631000,			
Total Stockholders' Equity	5425407000,	5389415000,	5563958000,	5783113000,
Accumulated other comprehensive income (loss), net	28695000,	54342000,	59194000,	29950000,
Common Stock paid-in-Capital-Shares				
Total Equity				
Total Liabilities and Equity	8310024000,	8339716000,	8492758000,	8991183000,

Fig. 23: Implementation of a balance sheet, using SPARK

For the FIOS 2.0 system we also included Pubby. However, no special changes were made for Pubby. We maintained the database URI prefix for Pubby according to the used base URI of the data in the system, called “http://public.b-kaempgen.de:8080/” . We mainly used Pubby for discovering the data and tried to find logical inconsistencies in the data model. Furthermore, it helped to verify if the integrated data is uploaded correctly.

Figure 24 illustrates the average net income of companies in different industry sectors. Another business question that could be solved, using Saiku and the OLAP functions would be how the net income loss changes for different industry sectors.

OLAP query, displaying the average net income loss for different SICs:

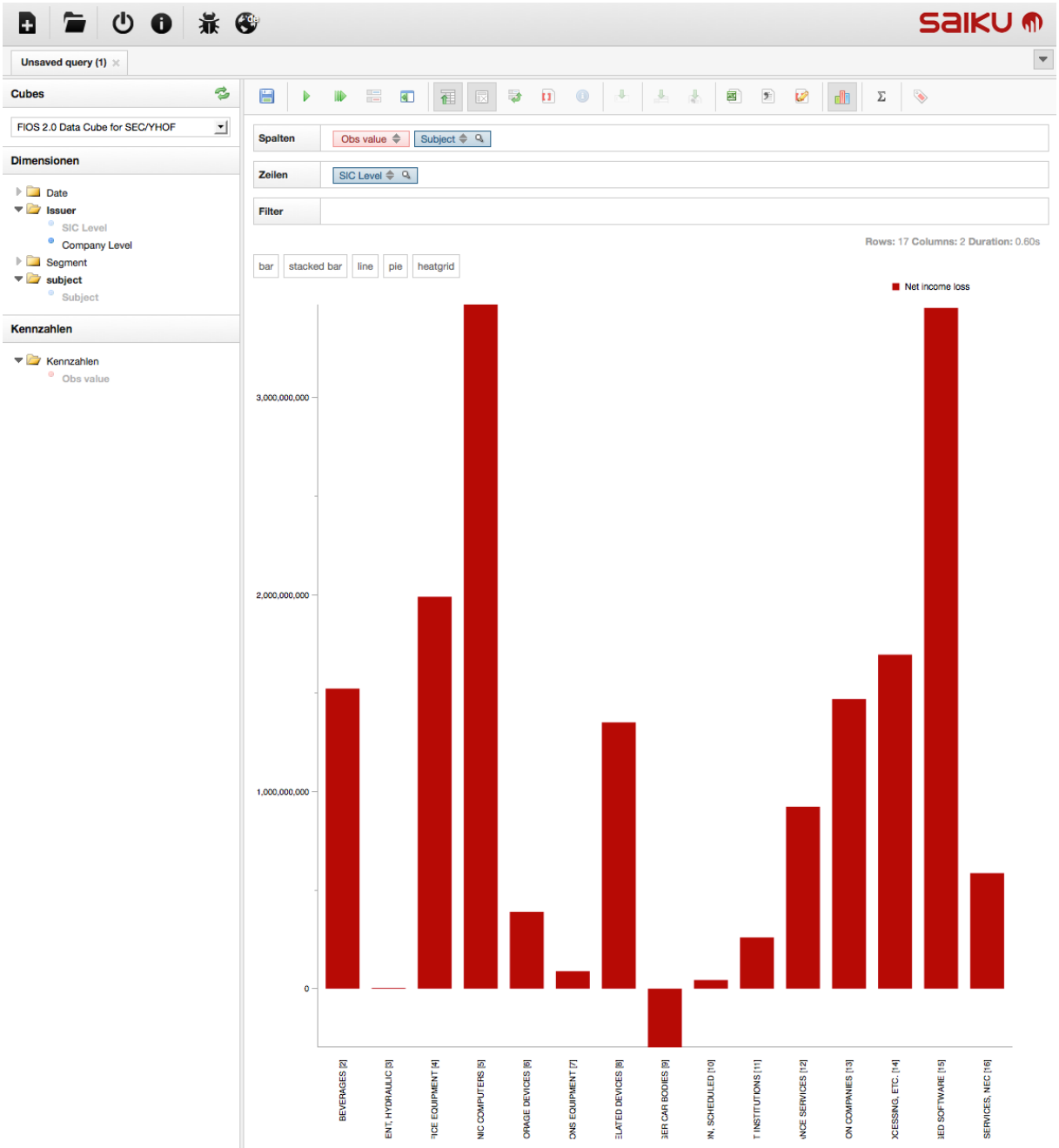


Fig. 24: Representing the average net income for different industry sectors using Saiku

4.3.2.3 Algorithm Analyzes

The algorithm analyzes were made for the normalizer and for the reasoner. The programs were executed for test purposes on a Windows 7 x86 PC, Genuine Intel® CPU U7300 @ 1.30 GHz, 4.00 GB and a Java Runtime Environment 7.0. We analyzed the runtime to see the performance of the program. The uploading process has not been analyzed due to lack of time. However, observations made from other people

show that 1 million triples can be uploaded in around 3-4 minutes.[39] The used system on which this observation is made is also an Ubuntu OS.

4.3.2.3.1 Normalizer

As introduced in chapter 4.2.2.3 - Normalization of URIs, different base URIs are normalized to one base URI, determined by the user. The runtime of this step in the ETL process can be seen in figure 25. The X-axis of the diagram shows the number of statements, which were tested. The Y-axis indicates in seconds the time needed to normalize the data file. We used one data file and increased gradually the number of triples in the file. We configured in the text file that 5 URIs should be changed. The measured time can be read of the diagram. Apparently, the process is linear. Due to the tests and the graph, we can say that the normalizer probably can be described linear in the big O notation. That is caused by the simple logic of the algorithm. The normalizer only cycles once through the data file and replaces the corresponding URIs in the text file. Due to the graph and test data it can be supposed that the algorithm is in a linear big O notation. A linear progress is a satisfying result. A possibility to make the normalizer faster is to arrange the URIs, read in the text file in a sorted array or a balanced search tree. However, we assume that this could only slightly improve the runtime.

Graph of the runtime results made for the Normalizer:

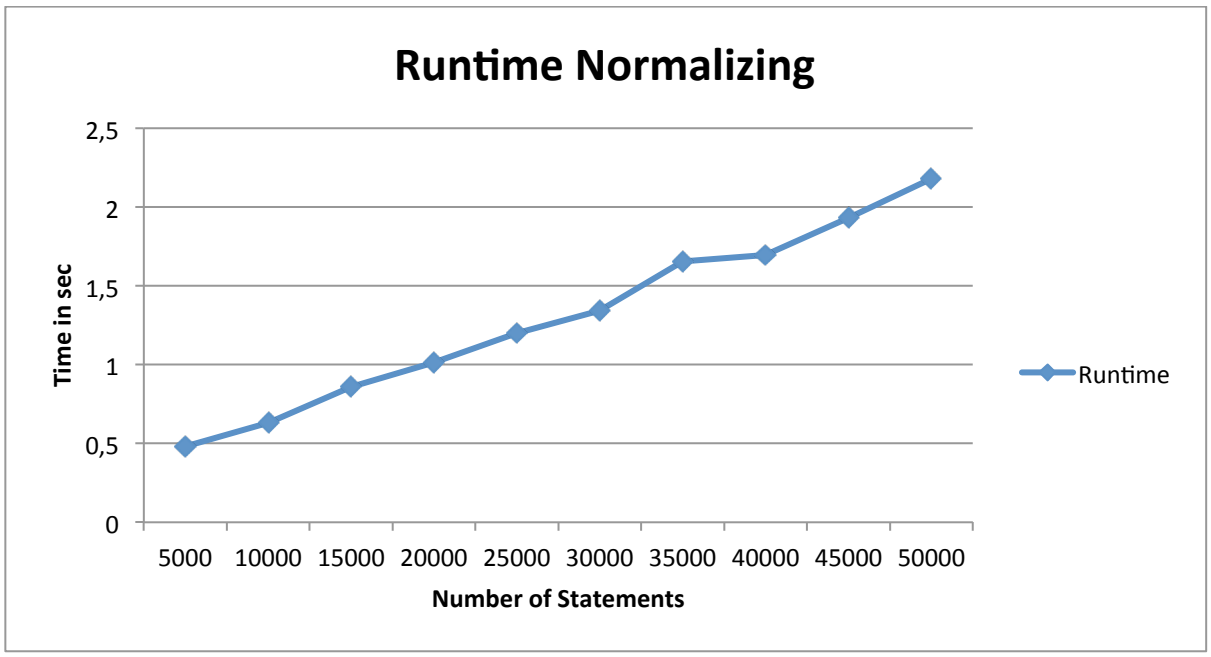


Fig. 25: Benchmark of the Normalizer

[39] http://boards.openlinksw.com/phpBB3/viewtopic.php?f=12&t=1174, 7th May 2013

4.3.2.3.2 Reasoner

The benchmarking of the reasoner contains the performance, monitored for different number of statements. We increased successively the number of statements in a triple file and used them for test purposes. We first analyzed only the time that is needed to map the owl:sameAs statements into the Hashtable. Thus we can see if the used structure and logic of the algorithm is sufficient. Then we analyzed as a further step the complete reasoning step, including the replacement of the resources.

Figure 26 shows the runtime of the reasoning regarding the number of owl:sameAs statements. The tested files contained only the owl:sameAs statements. And in this part of the analyzing, no replacing step was made. The Y axis indicates the time in seconds needed to execute the program. As you can see, the curve increases exponentially. From 5000 owl:sameAs statements, the runtime of the program increases rapidly. Due to the tests and the graph, we can say that the program probably can be described as exponential in the big O notation. However, the reasoning step for 10000 owl:sameAs statements can be done on the performed computer system in about 5 seconds. Though reasoning 10000 statements takes only about 5 seconds, the exponential growth of the runtime is not sufficient.

Graph with the test results of hashing the owl:sameAs statements:

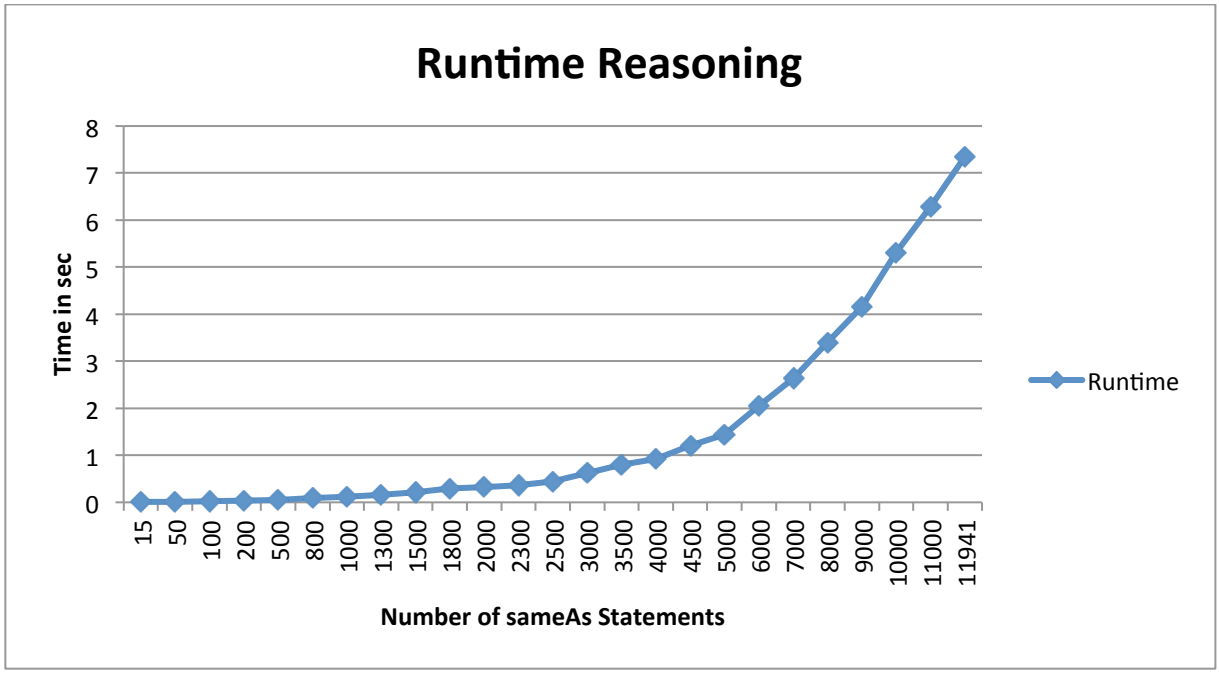


Fig. 26: Runtime of the reasoning step

The figure 27 also describes the runtime of the program, but regarding the owl:sameAs statements and the complete triples. Moreover, this part contains both, the reasoning step and the replacing step. The first number on the X axis shows the total statements in the file, the second below, shows the owl:sameAs statements. The Y axis shows the

time in seconds, needed to finish the reasoning and replacing step. As you can see, by executing also the replacing step after the reasoning step, the complexity does not change and is still exponential. At 291.670 statements or 2.317 owl:sameAs statements, the time needed for the execution increases steeply.

Graph of the test results made for the reasoning step:

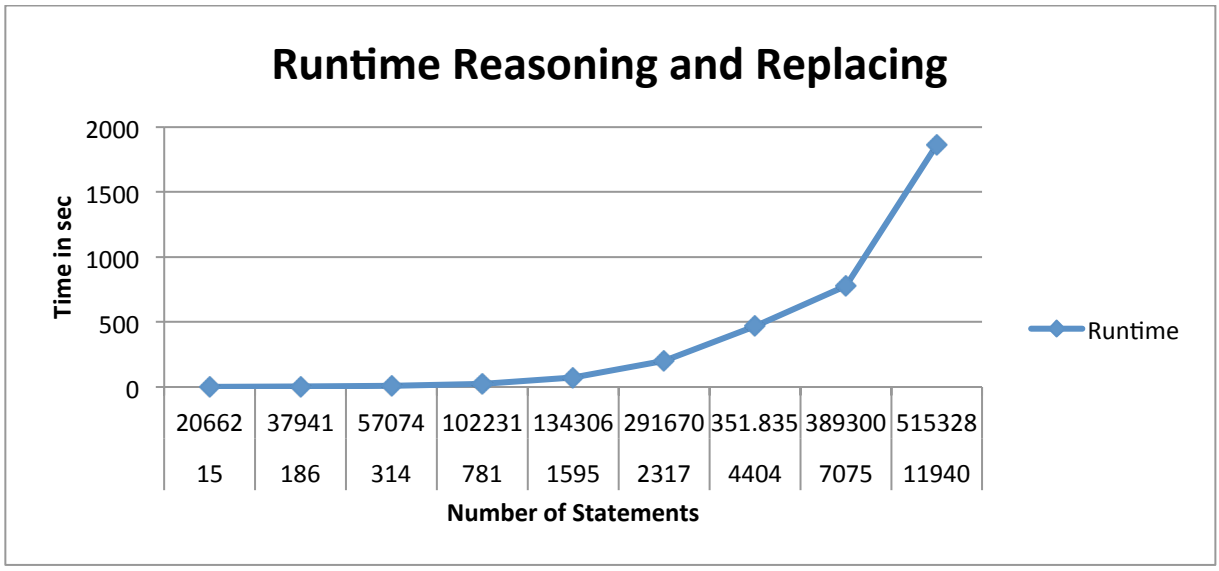


Fig. 27: Reasoning Runtime

As the results show, the reasoning step for the FIOS system is not sufficient. The exponential growth of the reasoner does not allow a satisfactory reasoning of RDF data files. Perhaps the performance of the system can be improved. So far the reasoner looks up every resource in the triple file at the Hashtable. If the linked lists inside of the Hashtable are very long, the application needs more time to check if the corresponding resource, which should replace the resource in the triple file, is available in the linked list. In the worst case, the resource is not available in the linked list and thus it will not be replaced. The process of checking every resource can be shortened by first reviewing whether the last resource that was replaced is the same resource as the current resource. Thus it will be replaced with the same entry of the Hashtable. As the result of the look up into the Hashtable is known, due to the last replacing step of the resource, no look up into the Hashtable is necessary. This shortens the runtime, but an exponential growth of the reasoning step cannot be prevented with this solution. The introduced solution only shifts the curve down, so less time is needed.

4.3.3 Requirement Coverage Analysis

This section deals with the question where the requirements made in chapter 4.1 - Requirements for Integrated Financial Analysis. We will see if the FIOS 2.0 system can answer the business questions and meet our expectations.

The integration of the Yahoo! Finance stock data as an additional data source succeeded, thus the FIOS 2.0 system can represent share stock value with SEC XBRL data. The system can represent the data, with the help of SPARK, in different forms, e.g. charts like pie chart or date chart or in tables. Besides the HTML representation of information, Saiku allows the querying over the data. To represent both, SEC XBRL and Yahoo! Finance data, as integrated data. Furthermore, year-to-year comparisons are possible. Year-to-year comparisons can either be done in Saiku, by querying over the corresponding data, or on the HTML page by considering the progress of key figures in the date chart. Users can themselves choose which illustration facilities they would like to use for considering their corresponding business questions. Therefore the three business questions in section 4.1.1 - Business Questions can be answered. Besides the comparison of years also comparisons between companies are possible. So different companies can be compared without restriction on a certain Industry sector. Comparisons of different business sectors are also possible due to the aggregation of the companies working in this sector. Furthermore, analyzes of the industry sector are possible over year-to-year. However, the representation of the data on a higher level than the SIC, e.g. the Major Group, is not possible. The induction of the Major Group and Division is not working so far. Thus analyzing on a higher aggregated level than the SIC is not possible. Also the analysis on country level is not possible, due to the fact that the country as a hierarchy level is missing in the data model. The data for the Major Group and the countries are not crawled and inducted into the FIOS system. However, we already wrote a wrapper, which provides data like the related country for companies.We called the Wrapper CompanyMetaWrapper[40] This wrapper also provides meta-information such as the number of inhabitants for the country or a description of the Major group. Thus we created the fundamental to enable this information in future for the FIOS system, to allow users to analyze the data according to the particular countries and major groups.

Also the usability of the system is comfortable due to the automation of the ETL process. Thus the system reduces efforts of accumulating the data by the user so business analysts can concentrate on analyzing the data due to the automated ETL process. Furthermore, the users do not need profound knowledge about the used data model or the data sources. These will be handled by the system. At some parts the user can engage the system and can e.g. add additional URIs for crawling the data or

[40] http://companymetawrap.appspot.com/, 7th May 2013

change the base URI of the normalization step. But if not, the system still works and provides data for business analysts.

4.3.4 Lessons Learned

We achieved objectives like the integration of an additional data source. This data source helps analysts to have more available data and therefore to consider different aspects of market progress on different views. Moreover, the automation of the ETL process takes care of the uploading of new data. Therefore, consequently keeping the data in the Open Virtuoso server actual provides the possibility to have always actual data available. However, we did not achieve the objective of representing the data on a higher aggregated level than the SIC due to lack of time. Closer to the date for submitting the application, we concentrated on improving the present hierarchy instead of including additional hierarchy levels to the industry group.

As explained in chapter 4.2.2.1 - Yahoo! Finance Wrapper there are inconsistencies in the archives. The Yahoo! Finance Wrapper retrieves all provided key figures when a client calls an archive on the actual date. However, calling the same archive one day later, only a few key figures will be retrieved through the Yahoo! Finance API does not provide all key figures historically. This inconsistency is a disadvantage, due to the fact that it does not provide a consistent number of key figures for all dates. Through the automated ETL process, which extracts everyday the stock data, we can deal with this solution. However, if a new company will be crawled, also for historic stock data, not the complete number of key figures are provided.

The analysis of the Normalizer retrieved satisfactory results, so the performance of this part of the ETL process is sufficient. However, the results of the Reasoner showed that this part of the ETL process contains improvement possibilities. Also the visualization could be improved by implementing the SIC hierarchy provided by our CompanyMetaWrapper. A further improvement could be the possibility to filter for multiple values in Saiku, so that more values can be displayed, which would lead to a more clearly representation of integrated data. Furthermore, a satisfactory solution for displaying balance sheet items in a balance sheet pattern on the HTML page with RDF linked data should be developed.

Unfortunately, we did not win the XBRL Challenge 2013 with the FIOS system. Even though the judges consider the FIOS system interesting, another application met the judges' criteria. According to David Tauriello, which is a person in charge in the XBRL US Consortium, the browser limitation was a setback. Despite that, the judges as well as we ourselves, believe in the strength of FIOS.

5 Conclusion

In this thesis we showed how we improved the FIOS system by connecting Linked Data from Yahoo! Finance as an additional data source to the open Data. Linking Yahoo! Finance data with SEC XBRL data and enriching them with data from other sources offers new possibilities for the data analysis. The chance to see coherences between data from different data sources, to evaluate XBRL data now within sectors and to compare them to historical data enables users to gather new conclusions, which are not possible when considering one period. The technical structure of the data sets is constructed in such a way that in future also hierarchies can be evaluated with SKOS. Furthermore, the developed programs can be implemented on every server, which can handle the java technology. Therefore, the application possibilities of the FIOS system, including the ETL process, are vast. The various settings enable a flexible usage of the system thus, beyond the financial data, also linked data. Moreover, the complete automation of the process provides actual data in a comfortable way. Although, other researches, as Niinimäki and Niemi did in 2009 [NiNi09], has also created an ETL process aiming at creating RDF data, we did a further step with the automation of the ETL process. In future, more transformation of sources like CSV or XML should follow into RDF and they should be published on the web. Thus the former isolated information in the non-RDF format is publicly accessible by people and machines, so that it can be turned to good account as knowledge can be derived from it.

We also achieved with this submission to the XBRL Challenge the integration of the Linked Data field of research into the financial sector. The XBRL Challenge helped us to promote the idea of an integrated system with an automated ETL process that provides data according to the Linked Data Principles. The feedback and results received through the XBRL Challenge can help us to improve the FIOS system in future.

Bibliography

[Bern06] Berners-Lee, T., *Linked Data*, 2006, http://www.w3.org/DesignIssues/LinkedData.html, Accessed on 7th May 2013.

[CLRS03] Cormen, T.H.; Leiserson C.E.; Rivest, R.L.; Stein, C.: *Introduction to Algorithms*, 2. Edition, Fourth printing, The Massachusetts Institute of Technology 2003, P. 226 f.

[Corm10] Cormen, T.H.: *Algorithmen – Eine Einführung*, 3. Edition, Oldenbourg Verlag, München 2010, P. 603 f.

[CyRe13] Cyganiak, R.; Reynolds, D.: The RDF Data Cube Vocabulary, 2013, http://www.w3.org/TR/vocab-data-cube/, 7th May 2013.

[DFOP09] Debreceny, R.; Felden, C.; Ochocki, B.; Piechocki, M.; Piechocki, M.: *XBRL for Interactive Data*, 1. Edition, Springer, Heidelberg 2009, Page 35; 52-53.

[Dyck00] Dyckhoff, R.: *Automated Reasoning with Analytic Tableaux and Related Methods*, Springer Verlag, Berlin, 2000.

[ElNa09] Elmasri, R.A.; Navathe, S.B.: *Grundlagen von Datenbanksystemen*, 3. Edition, Pearson Education Deutschland, München 2009, P. 40.

[FFST11] Fensel, D.; Facca, F. M.; Simperl, E.; Toma, I.: *Semantic Web Services*, 1. Edition, Springer Heidelberg 2011, P. 93.

[Fisc05] Fischer, E.O.: *Finanzwirtschaft für Anfänger*, 4. Edition, Oldenbourg Wissenschaftsverlag, München 2005, P. 155.

[Flan03] Flanagan, D.: Java in a Nutshell, 4. Edition, O'Reilly Verlag, Köln, 2003, P. 6.

[GABL13] Gabler Verlag, *Gabler Wirtschaftslexikon*, http://wirtschaftslexikon.gabler.de/Archiv/6782/finanzanalyse-v7.html.gabler.de/Archiv/6782/finanzanalyse-v7.html, Accessed on 7th May 2013.

[GeCh03] Geroimenko, V.; Chen, C.: *Visualizing the Semantic Web*, Springer, London 2003, P. 8 ff.

[GoMS12] Gomaa, M.I.; Markelevich, A.; Shaw, L.: *Introducing XBRL through a financial statement analysis project*, Suffolk University, Boston, 2012.

[Harb08] Harbich, R.: *Webcrawling – Die Erschließung des Webs*, 2008, http://www-e.uni-magdeburg.de/harbich/webcrawling/webcrawling.pdf, Accessed on 7th May 2013.

[HeBi11] Heath, T.; Bizer, C.: *Linked Data: Evolving the Web into a Global Data Space*, 1. Edition, Morgan & Claypool, 2011, P. 7 ff.

[HKRS08] Hitzler, P.; Krötzsch, M.; Rudolph, S.; Sure, Y.: *Semantic Web*, Springer, Berlin, 2008.

[HoWa09] Hoffman, C.; Watson, L.: *XBRL For Dummies*, 2009, http://www.dummies.com/how-to/content/xbrl-for-dummies-cheat-sheet.html, Accessed on 7th May 2013.

[IUBH13] Isele, R.; Umbruch, J.; Bizer, C.; Harth, A.: *LDSpider: An open-source crawling framework for the Web of Linked Data*, 2010, http://iswc2010.semanticweb.org/pdf/495.pdf, Accessed on 7th May 2013.

[KaBü08] Kastens, Prof. Dr. U.; Büning, Prof. Dr. H. K.: *Modellierung: Grundlagen und Formale Methode*, 2. Edition, Carl HanserVerlag, München 2008, P.215 f.

[KäHa12] Kämpgen, B.; Harth, A.: *Transforming Statistical Linked Data for Use in OLAP Systems*, 2012 http://www.aifb.kit.edu/images/2/28/Kaempgen_harth_isem11_olap.pdf, Accessed on 7th May 2013.

[KS09] Krüger, G.; Stark, T.: *Handbuch der Java Programmierung*, 6. Edition, Addison-Wesley, München 2009, P. 360.

[NiNi09] Niinimäki, M.; Niemi, T.: *An ETL Process for OLAP Using*

RDF/OWL Ontologies, 2009, http://link.springer.com/content/pdf/10.1007%2F978-3-642-03098-7_4.pdf, Accessed on 7th May 2013.

[Rect13] Rector, A.L.: *Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL*, 2003, http://www.cs.man.ac.uk/~rector/papers/rector-modularisation-kcap-2003-distrib.pdf, Accessed on 7th May 2013.

[SpSc10] Spermann, K.; Scheurle, P.: *Finanzanalyse*, Oldenburg Wissenschaftsverlag, München 2010, P. 8.

[StCa10] Stamey, J. W. Jr.; Casselman, C.: *Reporting and Sharing Financial Information with XBRL,* 2010, http://sais.aisnet.org/2010/2010-SAIS%20proceedings/Stamey-and-Casselman.pdf , Accessed on 7th May 2013.

[Trus13] Truscott, J.: *Umm Extensions…picking 3 companies at random*, 2013, http://www.blog.xbrlxl.com/, Accessed on 7th May 2013.

[USSE13] U.S. SEC, *The Investor's Advocate: How the SEC Protects Investors, Maintains Market Integrity, and Facilitates Capital Formation*, 2013, http://www.sec.gov/about/whatwedo.shtml, Accessed on 7th May 2013.

[Vass02] Vassiliadis, P.: *Modeling Multidimensional Databases, Cubes and Cube Operations,* 2002, http://infolab.usc.edu/csci599/Fall2002/paper/DML1_vassiliadis98modeling.pdf, 7th May 2013.

[WeWi08] Weverka, P.; Wilson, S. So.: *XBRL For Dummies*, Wiley Publishing, Indianapolis 2008, P. 23 f.

[Wood10] Wood, D.: *Linking Enterprise Data*, 1. Edition, Springer New York 2010, P. 103-125.

[YMYM10] Yoshikaea M.; Meng, X., Yumoto, T.; Ma, Q.; Sun, L.; Watanabe, C.: *Database Systems for Advanced Applications*, Springer, Berlin 2010, P. 45.